WHZ Westsächsische Hochschule Zwickau
Hochschule für Mobilität

Wissenschaftliche Arbeit

Ähnlichkeit von gesprochenen Zeichenketten

Ernaz Erkinbekov

Matrikel-Nr.: 55592
Studiengang Informatik
Betreuer: Prof. Dr. Laue

Fachgruppe Informatik
Fachbereich Physikalische Technik/ Informatik
Westsächsische Hochschule Zwickau

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

1.1 Einleitung

In den letzten Jahrzehnten haben das Problem der Verarbeitung natürlicher Sprache und insbesondere die Aufgaben im Zusammenhang mit der Analyse der phonetischen Ähnlichkeit von Wörtern immer mehr Aufmerksamkeit auf sich gezogen. Der wachsende Bedarf an hochpräzisen Spracherkennungs- und -verarbeitungssystemen, automatischer Übersetzung, Fehlerkorrektur und Phrasensuche hat dazu geführt, dass Softwarelösungen entwickelt werden müssen, die die Ähnlichkeit von Wörtern und Ausdrücken nicht nur anhand ihrer Schreibweise, sondern auch anhand ihres Klangs erkennen und berücksichtigen können.

Da ähnlich klingende Wörter völlig unterschiedliche Bedeutungen haben können, können Fehler bei ihrer Interpretation die Genauigkeit automatisierter Systeme erheblich reduzieren. Daher besteht ein Bedarf an Algorithmen, die nicht nur Zeichenketten auf Zeichenebene vergleichen, sondern auch deren phonetische Merkmale modellieren können. Beispiele solcher Algorithmen umfassen Soundex, Metaphone, Double Metaphone und Kölner Phonetik, die es ermöglichen, Wörter mit ähnlichem Klang zu identifizieren.

In dieser Übersicht sollen bestehende Algorithmen untersucht, ihre Fähigkeiten und Grenzen analysiert und ihre Effektivität beim Vergleich phonetisch ähnlicher Zeichenfolgen.

1.2 Ziel und Motivation

Das Ziel der Wissentschaftliches Arbeit ist es, eine Übersicht über bestehende Software/Algorithmen zu erstellen, die den Grad der Ahnlichkeit zwischen Zeichenketten effektiv messen können, indem sie sich auf deren Aussprache beziehen. Dabei wird besonders darauf geachtet, dass diese Werkzeuge kostenlos im Internet zugänglich sind und jeder diese frei verwenden kann. Auf diese Weise wird die Software/Algorithmen zur Messung der Ähnlichkeit von Zeichenketten erläutert, wobei sowohl ihre Vorteile als auch ihre Nachteile hervorgehoben werden.

2 Allgemeine Informationen zum Thema phonetische Änlichkeit

2.1 Was ist phonetische Ähnlichkeit?

Die phonetische Ähnlichkeit ist ein Maß dafür, wie ähnlich zwei Wörter oder Ausdrücke trotz möglicher Unterschiede in der Schreibweise klingen. Sie beachtet die Aussprache von Lauten und Silben sowie den Einfluss von phonetischen Veränderungen wie Akzenten, Dialekten und anderen Faktoren. Zweck der Bewertung der phonetischen Ähnlichkeit ist es, Wörter zu identifizieren, die gleich oder ähnlich klingen (z. B. Homophone), aber unterschiedlich geschrieben werden können. Homophone werden als Wörter definiert, die identische phonetische Eigenschaften aufweisen, sich jedoch in ihrer Schreibweise und Bedeutung unterscheiden. In der Studie von Ludovic Ferrand und Jonathan Grainger wird betont, dass Homophone wie z.B. MAID und MADE besondere Herausforderungen bei der visuellen Worterkennung darstellen, da sie einen Wettbewerb zwischen ihren phonologischen und orthografischen Repräsentationen erzeugen. Ludovic Ferrand, Jonathan Grainger, 2003

2.2 Begründung für die Wahl der Algorithmen

Die Algorithmen Soundex, Cologne Phonetic Metaphone und Double Metaphone wurden für die Untersuchung aufgrund ihrer Beliebtheit in Aufgaben des phonetischen Vergleichens von Zeichenketten ausgewählt. Ein zentraler Vorteil dieser Algorithmen liegt in ihrer einfachen Implementierung und hohen Berechnungseffizienz. Im Gegensatz zu komplexeren Methoden wie maschinellen Lernalgorithmen erfordern sie deutlich weniger Rechenressourcen, was sie besonders geeignet für reale Systeme macht, in denen eine schnelle Datenverarbeitung entscheidend ist, wie zum Beispiel Suchsysteme und automatische Fehlerkorrektur.

Diese Algorithmen haben den Vorteil, dass sie schnell eine binäre Antwort geben ("ähnlich" oder "unähnlich"), was den Vergleichsprozess vereinfacht und beschleunigt. Die phonetische Kodierung von Zeichenketten erleichtert die Analyse der Auswertung von Vergleichsergebnissen, was die Wahl der optimalen Methode für bestimmte Aufgaben und Sprachen ermöglicht.

2.3 Soundex Algorithmus

Der Soundex-Algorithmus ist ein Verfahren zur phonetischen Kodierung von Wörtern, das sicherstellt, dass ähnlich klingende Wörter denselben Code erhalten und so leichter identifiziert werden können. Ursprünglich 1918 von Robert Russell patentiert, wurde der Algorithmus speziell zur Indizierung von Namen in der Volkszählung entwickelt.

Wie Frankie Patman und Leonard Shaefer in ihrer Studie *"Is Soundex Good Enough for You? On the Hidden Risks of Soundex-Based Name Searching"* darlegen, basiert der Algorithmus darauf, dass Wörter in Großbuchstaben umgewandelt und Satzzeichen entfernt werden. Der erste Buchstabe bleibt unverändert, um den Klang des Wortes eindeutig zu identifizieren. Vokale (A, E, I, O, U) sowie die Halbvokale H, W und Y werden durch die Zahl "0" ersetzt, da sie für den phonetischen Code in der Mitte eines Wortes weniger relevant sind. Danach werden alle hinzugefügten "0"-Werte sowie identische aufeinanderfolgende Zahlen entfernt. F.Patman, L. Shaefer, 2003
Die Konsonanten werden entsprechend ihrer Lautgruppe in Zahlen umgewandelt:

Buchstabe	**Code**
B, F, P, V	1
C, G, J, K, Q, S, X, Z	2
D, T	3
L	4
M, N	5
R	6

Abbildung 1: Buchstabencodes des Soundex Algorithmus

Ist der resultierende Code kürzer als vier Zeichen, wird er mit Nullen aufgefüllt. Um eine Länge von vier Zeichen zu erreichen, wenn er langer ist, wird er gekürzt.

Um den Ablauf des Algorithmus zu veranschaulichen, wird der Name Ernaz“ als Beispiel betrachtet.

Buchstabe	**Code**	**Begründung**
E	bleibt erhalten	Der erste Buchstabe des Namens wird immer als Anfangsbuchstabe im Code übernommen.
R	6	Der Buchstabe **R** wird gemäß der Tabelle (siehe Abbildung 1) durch die Zahl 6 kodiert.
N	5	Der Buchstabe **N** wird gemäß der Tabelle durch die Zahl 5 kodiert.
A	—	Der Buchstabe **A** ist ein Vokal und wird im Soundex-Algorithmus ignoriert.
Z	2	Der Buchstabe **Z** wird gemäß der Tabelle durch die Zahl 2 kodiert.

Abbildung 2: Beispiel: Analyse des Namens mit dem Soundex-Algorithmus

Ergebniss: E652.

2.4 Kölner Phonetic Algorithmus

Die Kölner Phonetik wurde 1969 von Hans Joachim Postel entwickelt. Das Grundprinzip der Kölner Phonetik ist dem von Soundex ähnlich, jedoch wurden die Lauttabellen besser an die deutsche Sprache angepasst. Im Gegensatz zu Soundex wird nicht jedes Zeichen isoliert betrachtet, sondern es wird ein Kontext – bestehend aus maximal einem benachbarten Zeichen – betrachtet, um den tatsächlichen Laut dieses Zeichens genauer ermitteln zu können.

Buchstabe	Kontext	Code
A, E, I, J, O, U, Y		0
H		–
B		1
P	nicht vor H	1
D, T	nicht vor C, S, Z	2
F, V, W		3
P	vor H	3
G, K, Q		4
C	im Anlaut vor A, H, K, L, O, Q, R, U, X; vor A, H, K, O, Q, U, X außer nach S, Z	4
X	nicht nach C, K, Q	48
L		5
M, N		6
R		7
S, Z		8
C	nach S, Z	8
C	im Anlaut außer vor A, H, K, L, O, Q, R, U, X;	8
C	nicht vor A, H, K, O, Q, U, X	8
D, T	vor C, S, Z	8
X	nach C, K, Q	8

Abbildung 3: Buchstabencodes des Kölner Algorithmus

Ebenso gibt es für den Wortanfang (den sogenannten "Anlaut") teilweise spezielle Kodierungen. Abbildung 3 zeigt die Zuordnung der Buchstaben einer Lautgruppe zum jeweiligen Code der Kölner Phonetik. Christopher Deringer, 2024

Um besser zu verstehen, wie der Algorithmus funktioniert, hier ein Beispiel für "Wissenschaftliche Arbeit". Zunächst werden alle Buchstaben in Großbuchstaben umgewandelt, und jedem Buchstaben wird ein numerischer Code zugewiesen.

Buchstabe	Grund	Code
W	kodiert als 3.	3
I	wird ignoriert (Vokal).	-
S	kodiert als 8.	8
S	kodiert als 8, wird aber ignoriert, weil er nach einem ähnlichen Code kommt.	-
E	wird ignoriert (Vokal).	-
N	kodiert als 6.	6
S	kodiert als 8.	8
C	kodiert als 8, wird aber ignoriert, weil der vorherige Code 8 ist.	-
H	wird ignoriert.	-
A	wird ignoriert. (Vokal)	
F	kodiert als 3.	3
T	kodiert als 2.	2
L	kodiert als 5.	5
I	wird ignoriert (Vokal).	-
C	kodiert als 4 (vor H).	4
H	wird ignoriert.	-
E	wird ignoriert. (Vokal)	-
A	wird ignoriert (Vokal).	-
R	kodiert als 7.	7
B	kodiert als 1.	1
E	ignoriert (Vokal).	-
I	ignoriert. (Vokal)	-
T	kodiert als 2.	2

Abbildung 4: "Wissenschaftliche Arbeit" als Beispiel für Kölner Algorithmus

Ergebniss: 38683254712

2.5 Metaphone Algorithmus

Laut dem Buch *Practical Algorithms for Programmers*, das von Binstock und Rex herausgegeben wurde, verwendet der von Lawrence Philips entwickelte Algorithmus Metaphone Regeln für die präzise Umwandlung von Lauten in Wörter. Im Gegensatz zu Soundex berücksichtigt Metaphone Buchstabengruppen (Diphthonge) und entfernt nicht-alphabetische Zeichen aus dem Suchschlüssel, bevor die verbleibenden Buchstaben in Großbuchstaben umgewandelt werden (siehe Abbildung 5). Alle Vokale, außer dem ersten (falls das Wort mit einem Vokal beginnt), werden ausgeschlossen.
Die Methode arbeitet mit komplexeren Regeln für Konsonanten und Buchstabenkombinationen. Zum Beispiel wird "PH" zu "F" und "Q" zu "K" umgewandelt. Im Gegensatz zu Soundex, das besondere Fälle nicht erkennt, verarbeitet Metaphone diese mithilfe zusätzlicher Transformationen, was es in einigen Fällen beim Vergleich präziser macht. Binstock, A., Rex. J, 1995

Von	**Zu**
GN-, KN-, PN-	N
AE-	E
WH-	H
WR-	R
X-	S
B	B (außer in -MB)
C	X, wenn in -CIA-, -CH- S, wenn in -CI-, -CE-, -CY- entfällt, wenn in -SCI-, SCE-, -SCY- sonst K
D	J, wenn in -DGE-, -DGI-, oder -DGY- sonst T
G	F, wenn in -GH und nicht B–GH, D–GH, -H–GH, -H—GH entfällt, wenn in -GNED, -GN, -DGE-, -DGI-, oder -DGY- J, wenn in -GE-, -GI-, -GY- und nicht GG sonst K
H	H, wenn vor einem Vokal und nicht nach C, G, P, entfällt, wenn nach C
K	entfällt, wenn nach C, sonst K
P	F, wenn vor H, sonst P
Q	K
S	X, wenn in -SIO- oder -SIA-, sonst S
T	X, wenn in -TIA- oder -TIO- O vor H sonst T
V	F
W	W nach einem Vokal, sonst entfällt
X	KS
Y	Y, wenn nicht gefolgt von einem Vokal
Z	S
F, J, L, M, N, R	unverändert

Abbildung 5: Regeln für die Umwandlung von Buchstaben im Metaphone-Algorithmus

Um zu verstehen, wie der Algorithmus funktioniert, wird das Wort "Phonetik" im Abbildung 6 analysiert.

Buchstabe	Regel	Code
P	Wenn der Buchstabe P“ vor ”H” steht (wie in ”PH”), wird er in ”F” umgewandelt.	F
H	Nach ”PH” wird der Buchstabe ”H” nicht mehr berücksichtigt, da er bereits zusammen mit ”P” verarbeitet wurde.	—
O	Vokal, der nicht am Wortanfang steht, daher wird er ignoriert.	—
N	Bleibt erhalten, da N“ sich nicht verändert.	N
E	Vokal, wird ignoriert.	—
T	Bleibt erhalten, da T“ sich nicht verändert.	T
I	Vokal, wird ignoriert.	—
K	Bleibt erhalten, da K“ sich nicht verändert.	K

Abbildung 6: ”Phonetik” als Beispiel für Metaphone Algorithmus.

Ergebniss: FNTK

2.6 Double Metaphone Algorithmus

Gemäß dem Artikel ”Phonetic Matching: A Better Soundex” im Association of Professional Genealogists Quarterly, verfasst von Alexander Beider und Stephen P. Morse, wurde der Double Metaphone-Algorithmus von Lawrence Philips als Weiterentwicklung des ursprünglichen Metaphone konzipiert. Der wichtigste Unterschied liegt darin, dass Double Metaphone für ein Wort zwei Kodierungen erzeugt: eine primäre und eine alternative. Dies macht ihn deutlich robuster als Metaphone, das nur eine einzige Kodierung liefert. Zudem berücksichtigt Double Metaphone fremdsprachige Aussprachen, indem er Regeln für unterschiedliche Sprachen integriert, jedoch ohne die spezifische Zuordnung zu einem bestimmten Sprachsystem. Ein weiterer Unterschied besteht darin, dass Double Metaphone wieder auf eine beschränkte Kodierung des Namensanfangs zurückgreift, ein Konzept, das Metaphone ursprünglich zu erweitern versuchte. Diese Einschränkung macht Double Metaphone jedoch genauer bei der Behandlung von Namen und Schreibvarianten, insbesondere in multikulturellen Kontexten. Der Unterschied zwischen Metaphone und Double Metaphone Algorithmen wird in Kapitel **4.2** behandelt. A. Beider, Stephen Morse, März 2010

Buchstabe	**Code(s)**	**Kontext**
A, E, I, O, U, Y	A	Nur am Wortanfang.
K,M,L,N,R	K,M,L,N,R	immer
B	P	immer
V,F	F	immer
Q	K	immer
C	K, S, X	K: vor ACH ohne nachfolgende Vokale. S: vor CE, CI, CY. X: vor CH. KS: wenn CC vor I, E, H steht.
D	T, J	J: wenn GE, GI, GY folgt. T: sonst.
G	K, J, (leer)	K: ohne nachfolgende Vokale oder vor GH. J: vor GI, GE, GY. Leer: vor GH nach B, H, D.
H	H, (leer)	H: am Wortanfang oder zwischen zwei Vokalen. Leer: sonst.
J	J, H	H: in spanischen Wörtern (z.B. "José"). J: sonst.
P	P, F	F: wenn P vor H steht. P: sonst.
S	S, X	X: vor SH oder in italienischen Wörtern. S: vor SC vor I, E, Y.
T	T, X, 0	X: vor TION, TIA, TCH. 0: vor TH (außer in deutschen Wörtern). T: sonst.
W	A, F, (leer)	A: am Wortanfang vor einem Vokal. F: in der Mitte oder am Wortende. Leer: sonst.
X	KS	Immer, außer am Ende eines französischen Wortes (leer).
Z	S, TS	TS: in slawischen Wörtern. S: sonst.

Abbildung 7: Regeln der Buchstabenkodierung im Double Metaphone

Die Analyse des Double Metaphone-Algorithmus erfolgt ähnlich wie beim ursprünglichen Metaphone. Allerdings sollten dabei die zusätzlichen Buchstabenkombinationen und deren Umwandlungen berücksichtigt werden, die in Abbilding 7 dargestellt sind. Diese Regeln umfassen die Behandlung spezieller Fälle wie SCH“, CZ“, GN“ sowie die Berücksichtigung der phonetischen Besonderheiten verschiedener Sprachen.

3 Uberblick über vorhandene Software

Nachfolgend finden Sie eine Übersicht über verschiedene frei verfügbare und online verfügbare Vergleichswerkzeuge für Zeichenketten. Die Werkzeuge sind in verschiedene Bereiche unterteilt. Ein Teil besteht aus webbasierten Anwendungen, die über einen Browser und ohne Herunterladen genutzt werden können. Der andere Teil ist eine Gruppe von Werkzeugen, die aus vorgefertigten Bibliotheken für verschiedene Programmiersprachen besteht, die in den eigenen Programmiercode integriert und nach den eigenen Vorstellungen verwendet werden können. Zuvor wird jedoch ein kurzer Überblick über die grundlegenden Anpassungsmethoden gegeben.

3.1 Strategie zur Erstellung der Übersicht

Zu Beginn der Erforschung wurden zunächst verschiedene Ergebnisse der Standardsuchmaschine Google“ systematisch überprüft, um die folgende Übersicht zu erstellen. Die Suchanfragen wurden gezielt auf Begriffe ausgerichtet, die sich mit phonetischer Ähnlichkeit und entsprechenden Algorithmen beschäftigen. Folgende Begriffsgruppen wurden untersucht: *Phonetische Ähnlichkeit Werkzeuge, Phonetische Ähnlichkeit Algorithmen, Soundex online, Kölner Phonetik online, Metaphone online, Double Metaphone.* Da es verschiedene Homophone gab, um die Software zu testen, wurde nach Homophonen auf Deutsch gesucht – *Deutsche Homophonen.*
Es wurden deutsch- und englischsprachige Seiten identifiziert, die Werkzeuge für den Vergleich von Zeichenketten auf der Grundlage klanglicher Ähnlichkeit anbieten.

Bei der Suche wurde festgestellt, dass für den Zeichenkettenvergleich überwiegend Webanwendungen oder Bibliotheken für verschiedene Programmiersprachen verfügbar sind. Es erwies sich als herausfordernd, Anwendungen zu finden, deren Algorithmen speziell auf die Erkennung deutscher Wörter abgestimmt sind, im Gegensatz zu solchen, die primär für die Kodierung englischer Wörter entwickelt wurden. Nach gründlicher Recherche konnten jedoch mehrere Websites mit Algorithmen identifiziert werden, die auf den Vergleich deutscher Wörter ausgelegt sind.

3.2 Webanwendung: Tilores

Wenn es darum geht, Zeichenketten schnell und einfach auf phonetische Ähnlichkeit zu vergleichen, erweisen sich Online-Dienste oft als hilfreiches Werkzeug. Solche Dienste lassen sich leicht über Google oder andere Suchmaschinen finden und zeichnen sich in der Regel durch ihre Benutzerfreundlichkeit und intuitive Bedienung aus. Ein weiterer Vorteil ist, dass keine Installation oder der Download von spezieller Software erforderlich ist.

Ein Beispiel für den Vergleich von Zeichenketten ist die Webanwendung Tilores. Es bietet verschiedene Algorithmen für den Vergleich von Wörtern auf phonetische Ähnlichkeit, darunter Soundex, Kölner Phonetik und Metaphone. Das Werkzeug überzeugt durch seine intuitive Bedienung und eine übersichtliche Benutzeroberfläche. Die Anwendung stellt zwei Eingabefelder bereit, in die Wörter geschrieben oder eingefügt werden können. Anschließend liefert die Seite ein klares Ergebnis: true oder false. Darüber hinaus werden die kodierten Versionen der eingegebenen Wörter angezeigt, was eine bessere Nachvollziehbarkeit der Ergebnisse ermöglicht. Diese kodierten Ausgaben entsprechen den Beispielen, die in den Kapiteln zur Definition der Algorithmen erläutert wurden (siehe Verweise:
Soundex: Abbildung 2; Kölner: Abbildung 3; Metaphone: Abbildung 6). Ein weiterer Vorteil dieser Seite ist, dass sie einen Link zu den zu vergleichenden Zeichenketten erzeugt.

Nachfolgend sind Abbildungen dargestellt, die die Arbeitsweise der einzelnen Algorithmen anhand ihrer Schnittstellen zeigen, wie sie von der Tilores-Webanwendung bereitgestellt werden.

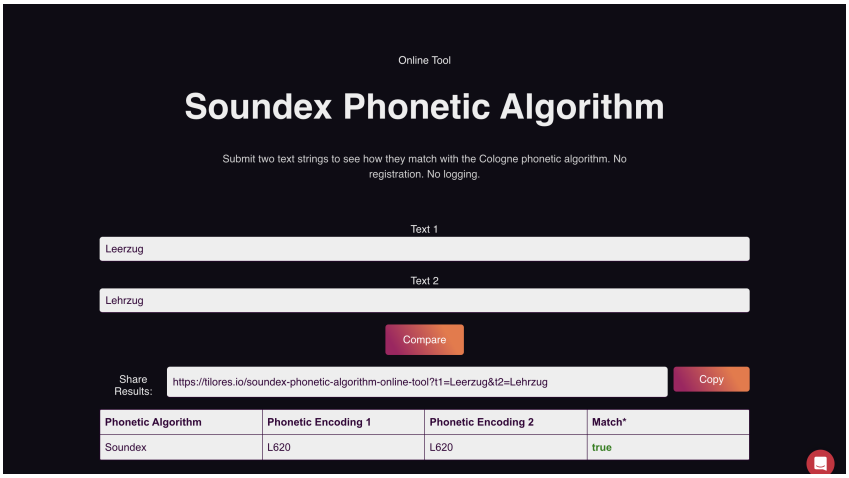

Abbildung 8: Zeichenkettenvergleich mit dem Soundex Algorithmus in der Tilores-Webanwendung

Nach der Auswahl des Soundex Algorithmus und der Eingabe der Wörter *Leerzug* und *Lehrzug* wurde ein Link zur Vergleichsanalyse generiert: https://tilores.io/soundex-phonetic-algorithm-online-tool?t1=Leerzugt2=Lehrzug. Auf der Seite wird eine Tabelle mit dem Namen des Algorithmus und den folgenden Daten angezeigt:

1. Phonetischer Code des ersten Wortes (Leerzug) — **L620**.
2. Phonetischer Code des zweiten Wortes (Lehrzug) — **L620**.

Das Vergleichsergebnis (*Match**) lautet **true**, was die phonetische Ähnlichkeit der Zeichenketten bestätigt.

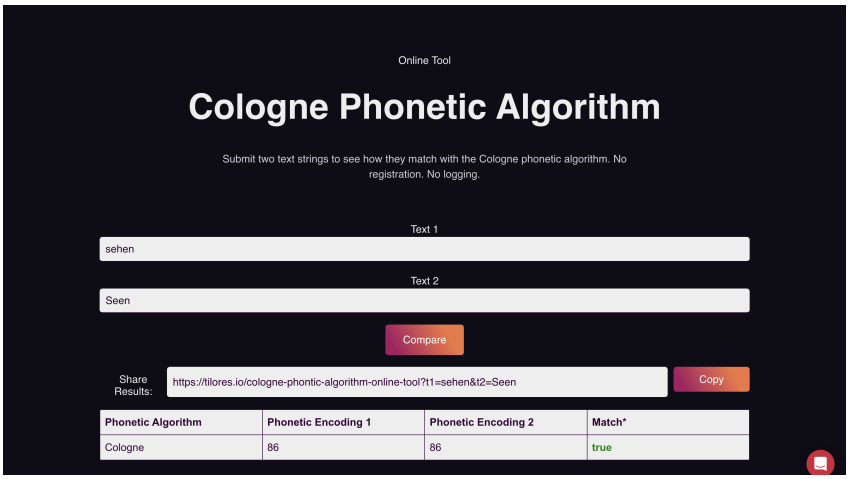

Abbildung 9: Zeichenkettenvergleich mit dem Kölner Algorithmus in der Tilores-Webanwendung

Nach der Auswahl des Kölner Algorithmus und der Eingabe der Wörter *sehen* und *Seen* wurde ein Link zur Vergleichsanalyse generiert: https://tilores.io/cologne-phontic-algorithm-online-tool?t1=sehent2=Seen. Auf der Seite wird eine Tabelle mit dem Namen des Algorithmus und den folgenden Daten angezeigt:

1. Phonetischer Code des ersten Wortes (sehen) — **86**.
2. Phonetischer Code des zweiten Wortes (Seen) — **86**.

Das Vergleichsergebnis (*Match**) lautet **true**, was die phonetische Ähnlichkeit der Zeichenketten bestätigt.

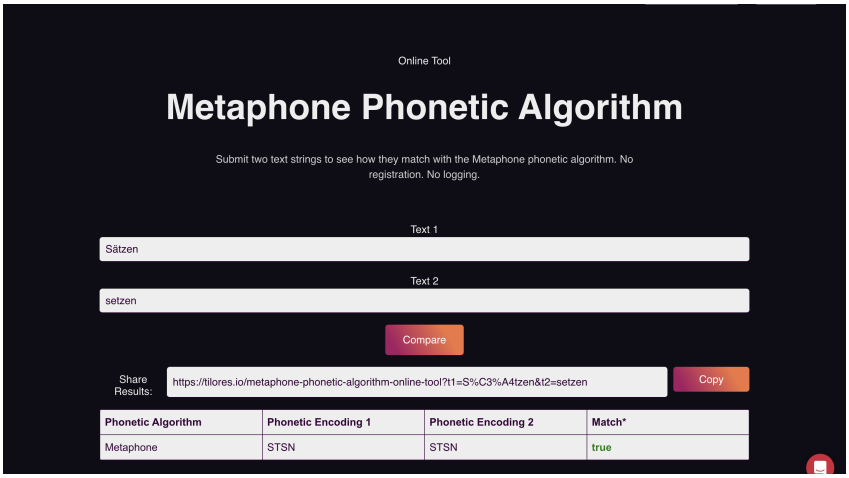

Abbildung 10: Zeichenkettenvergleich mit dem Metaphone Algorithmus in der Tilores-Webanwendung

Nach der Auswahl des Metaphone Algorithmus und der Eingabe der Wörter *Sätzen* und *setzen* wurde ein Link zur Vergleichsanalyse generiert: https://tilores.io/metaphone-phonetic-algorithm-online-tool?t1=A4tzent2=setzen. Auf der Seite wird eine Tabelle mit dem Namen des Algorithmus und den folgenden Daten angezeigt:

1. Phonetischer Code des ersten Wortes (Sätzen) — **STSN**.
2. Phonetischer Code des zweiten Wortes (setzen) — **STSN**.

Das Vergleichsergebnis (*Match**) lautet **true**, was die phonetische Ähnlichkeit der Zeichenketten bestätigt.

3.3 Webanwendung: Genealogy Online

Die Website Genealogy Online bietet die Nutzung der Algorithmen Soundex, Metaphone und Double Metaphone an. Der Hauptunterschied zu Tilores besteht darin, dass die Ergebnisse gleichzeitig für alle drei Algorithmen angezeigt werden und zusätzlich phonetisch ähnliche Begriffe vorgeschlagen werden.

Da diese Algorithmen hauptsächlich für die Suche nach phonetischer Ähnlichkeit von Namen entwickelt wurden, ermöglicht die Website die Erstellung einer Liste ähnlicher Namen basierend auf dem eingegebenen Wort. Obwohl es nicht möglich ist, Zeichenketten direkt zu vergleichen, bietet die Anwendung die Möglichkeit, die Kodierung von Wörtern zu erhalten, was bereits eine nützliche Funktion darstellt.

Ein weiterer Vorteil von Genealogy Online ist die Verfügbarkeit der deutschen Sprache. Hervorzuheben ist die Implementierung des Double Metaphone Algorithmus, der im Vergleich zum einfachen Metaphone besser an die deutsche Sprache angepasst ist. Aus diesem Grund wurde diese Website für die Analyse ausgewählt. ist eine Abbildung dargestellt, die die Schnittstelle der Algorithmen Soundex, Metaphone und Double Metaphone zeigt, wie sie von der Webanwendung Genealogy Online bereitgestellt werden.

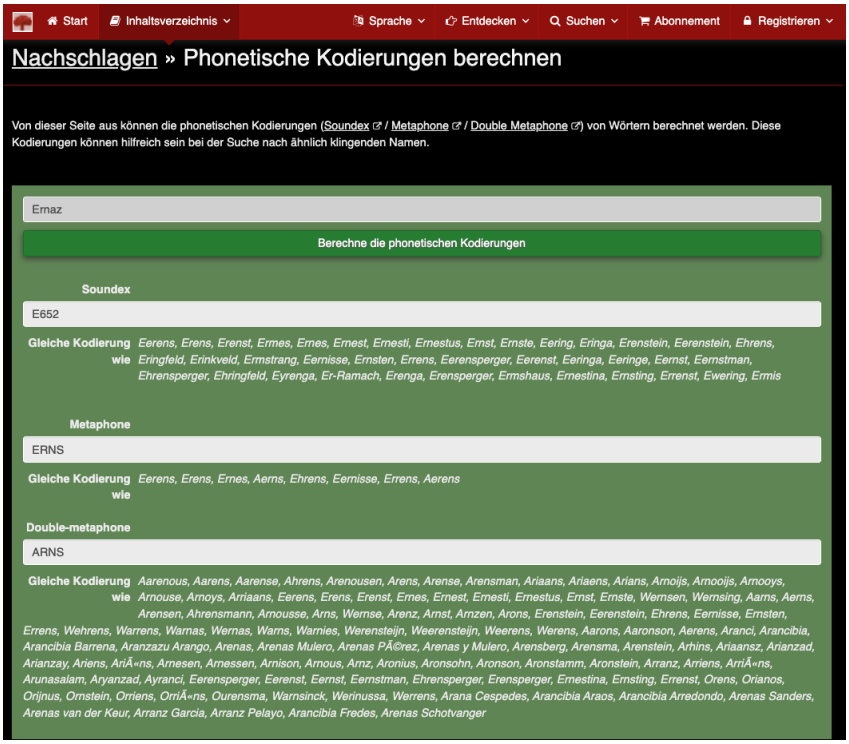

Abbildung 11: Codierung des Namens "Ernaz" mit verschiedenen Algorithmen bei Genealogy Online

Nach der Eingabe einer Zeichenkette (in diesem Fall des Namens *Ernaz*) werden die Ergebnisse der drei Algorithmen angezeigt:

- **Soundex**: E652,
- **Metaphone**: ERNS,
- **Double Metaphone**: ARNS.

Unter jedem Algorithmus werden alternative phonetische Varianten der Zeichenkette angezeigt. Es ist jedoch zu beachten, dass diese Funktionalität zur Website *Genealogy Online* gehört und nicht Teil der Algorithmen selbst ist (Double Metaphone kann eine zweite Option hervorbringen A. Beider, Stephen Morse, März 2010).

Bereits an dieser Stelle werden Unterschiede zwischen den Ergebnissen von Metaphone und Double Metaphone deutlich. Ein ausführlicher Vergleich der Algorithmen wird in Kapitel 4 behandelt.

3.4 Bibliotheken für phonetische Algorithmen in Python

Wie bereits erwähnt, gibt es zahlreiche Bibliotheken, die Algorithmen zur phonetischen Ähnlichkeit für verschiedene Programmiersprachen bereitstellen. Für die Programmiersprache Python stehen beispielsweise die Open-Source-Bibliotheken Phonetisch, Cologne-Phonetics und Metaphone zur Verfügung. Diese können für die Analyse und den Vergleich von Zeichenketten basierend auf ihrer phonetischen Ähnlichkeit verwendet werden.

- Die Bibliothek **Phonetisch** implementiert den Soundex-Algorithmus.
- Die Bibliothek **Cologne-Phonetics** bietet die Kölner Phonetik.
- Die Bibliothek **Metaphone** enthält den Algorithmus Double Metaphone.

Diese Bibliotheken bieten Entwicklern eine bequeme Möglichkeit, Algorithmen zur phonetischen Ähnlichkeit in ihre Projekte zu integrieren.

Um diese Bibliotheken zu verwenden, muss Python auf dem Computer installiert sein. Die Installation

erfolgt über das Terminal mit dem Befehl `pip`. Vergewissern Sie sich jedoch vorher, dass Sie sich in Ihrem Projektordner befinden und die *Virtual enviroment* aktiviert ist. Wie man *Virtual enviroment* erstellt und aktiviert, kann man hier nachlesen.

```
~/PycharmProjects/DoubleMetaphone DoubleMetaphone ❱ pip install phonetisch
pip install cologne-phonetics
pip install Metaphone
```

Abbildung 12: Laden der fertigen Bibliotheken für Soundex, Double Metaphone, Kölner Algorithmen mit dem Befehl `pip`

Um die heruntergeladenen Bibliotheken einzubinden, müssen die entsprechenden Module mit dem Befehl `import` importiert werden. Die Verwendung von Aliasnamen mit dem Schlüsselwort `as` macht den Code lesbarer und ermöglicht es, die verwendeten Algorithmen klar zu benennen.

```
from metaphone import doublemetaphone as doubleMetaphoneAlgorithmus
from phonetisch import soundex as soundexAlgorithmus
import cologne_phonetics as cologneAlgorithmus
```

Abbildung 13: Einbindung von Bibliotheken in den Code mit Hilfe des Befehls `import`

Die Wörter, die für die Analyse von Webanwendungen verwendet wurden, dienen ebenfalls als Beispiele für die Testung von fertigen Bibliotheken in der Programmiersprache Python. Jede Bibliothek bietet Funktionen zur Kodierung und Analyse von Zeichenketten gemäß dem ausgewählten Algorithmus.

Zur Kodierung und zum Vergleich der Wörter *leerzug* und *lehrzug* mit dem Soundex-Algorithmus wird die Methode `encode_word` aufgerufen, der das gewünschte Wort übergeben wird. Ein Beispiel ist in der Abbildung 14 dargestellt.

```
6  leerzug = soundexAlgorithmus.encode_word("Leerzug")
7  lehrzug = soundexAlgorithmus.encode_word("Lehrzug")
8  print("Leerzug:", leerzug, "\nLehrzug:", lehrzug, "\nPhonetische Änlichkeit ist:", lehrzug==leerzug)
9
```

Abbildung 14: Kodierung mit Soundex (Bibliothek phonetisch).

Zur Kodierung der Wörter *sehen* und *seen* mit dem Algorithmus Kölner Phonetik wird die Methode `encode` verwendet. Das gewünschte Wort wird dabei als Parameter übergeben. Ein Beispiel ist in der Abbildung 15 dargestellt.

```
11 sehen = cologneAlgorithmus.encode("sehen")
12 seen = cologneAlgorithmus.encode("Seen")
13 print("sehen:", sehen, "\nSeen:", seen, "\nPhonetische Änlichkeit ist:", seen[0][1]==sehen[0][1])
```

Abbildung 15: Kodierung mit Kölner Phonetik (Bibliothek cologne-phonetics).

Double Metaphone (Bibliothek metaphone). Zur Kodierung der Wörter *Sätzen* und *setzen* mit dem Algorithmus Double Metaphone wird die Methode `doublemetaphone` verwendet. Diese Methode nimmt eine Zeichenkette als Parameter und gibt ein Tupel mit zwei Elementen (primärer und alternativer Code) zurück. Ein Beispiel ist in der Abbildung 16 dargestellt.

```
16 satzen = doubleMetaphoneAlgorithmus("Sätzen")
17 setzen = doubleMetaphoneAlgorithmus("setzen")
18 print("Sätzen:", satzen, "\nsetzen:", setzen, "\nPhonetische Änlichkeit ist:", satzen[0]==setzen[0])
```

Abbildung 16: Kodierung mit Double Metaphone (Bibliothek metaphone).

Für den Vergleich der Ergebnisse in den Zeilen 13 und 18 wurden Indizes verwendet, da die Methode `encode` eine Liste zurückgibt, die Tupel mit zwei Elementen enthält. Auch die Methode `doublemetaphone` gibt ein Tupel zurück, was bei der Ergebnisverarbeitung berücksichtigt werden muss.

Der vollständige Code kann hier eingesehen werden:

```python
from metaphone import doublemetaphone as doubleMetaphoneAlgorithmus
from phonetisch import soundex as soundexAlgorithmus
import cologne_phonetics as cologneAlgorithmus

print("\nSOUNDEX")
leerzug = soundexAlgorithmus.encode_word("Leerzug")
lehrzug = soundexAlgorithmus.encode_word("Lehrzug")
print("Leerzug:", leerzug, "\nLehrzug:", lehrzug, "\nPhonetische Änlichkeit ist:", lehrzug==leerzug)

print("\nKÖLNER")
sehen = cologneAlgorithmus.encode("sehen")
seen = cologneAlgorithmus.encode("Seen")
print("sehen:", sehen, "\nSeen:", seen, "\nPhonetische Änlichkeit ist:", seen[0][1]==sehen[0][1])

print("\nDOUBLE METAPHONE")
satzen = doubleMetaphoneAlgorithmus("Sätzen")
setzen = doubleMetaphoneAlgorithmus("setzen")
print("Sätzen:", satzen, "\nsetzen:", setzen, "\nPhonetische Änlichkeit ist:", satzen[0]==setzen[0])
```

Abbildung 17: Ganzer Code

Wenn man den Python-Code ausführt, erhält man das folgende Ergebnis:

```
SOUNDEX
Leerzug: L622
Lehrzug: L622
Phonetische Änlichkeit ist: True

KÖLNER
sehen: [('sehen', '86')]
Seen: [('seen', '86')]
Phonetische Änlichkeit ist: True

DOUBLE METAPHONE
Sätzen: ('STSN', '')
setzen: ('STSN', '')
Phonetische Änlichkeit ist: True

Process finished with exit code 0
```

Abbildung 18: Ergebnis der vorgefertigten Bibliotheken in Python

Die Analyse der Ergebnisse zeigt eine erfolgreiche Demonstration der Nutzung von Open-Source-Bibliotheken in Python, um Zeichenketten basierend auf ihrer phonetischen Ähnlichkeit zu vergleichen. Es gibt jedoch einige Bedenken hinsichtlich der Tests:

- Nicht alle möglichen Fälle (z. B. Dialekte oder ungewöhnliche Schreibweisen) wurden möglicherweise abgedeckt.
- Die Effizienz der Methoden kann von der Sprache oder den Daten abhängen.

4 Vergleichende Analyse

In diesem Kapitel werden folgende Aspekte berücksichtigt:

- Der Vergleich der Algorithmen im Hinblick auf die Qualität der berechneten phonetischen Ähnlichkeit.
- Die Analyse der Algorithmen in verschiedenen Anwendungsszenarien, beispielsweise: Unter welchen Bedingungen ist ein Algorithmus einem anderen vorzuziehen?
- Die Ergebnisse der Algorithmen bei der Anwendung auf deutsche und russische Wörter.

4.1 Strategie zur Durchführung von Vergleichen

Zur Untersuchung werden Paare von Homophonen aus der Seite Wiktionary ausgewählt. Jedes Homophonpaar wird mithilfe verschiedener Algorithmen kodiert und verglichen. Wenn ein Algorithmus keine phonetische Ähnlichkeit zwischen Wörtern erkennt, werden die Ursachen für dieses Ergebnis analysiert und die Einschränkungen des jeweiligen Algorithmus untersucht. Abschließend werden die Vorteile anderer Algorithmen aufgezeigt, die dazu beitragen, die identifizierten Einschränkungen zu überwinden.

Für den Algorithmus Double Metaphone ist ein spezieller Ansatz erforderlich, da keine Webanwendung für den direkten Vergleich von zwei Zeichenketten existiert. Zwei Lösungen werden verfolgt:

- Kodierung der Zeichenketten einzeln mit der Website Genealogy Online und anschließender manueller Vergleich der resultierenden Codes
- Anwendung der Methode `doublemetaphone` aus der Python Bibliothek `metaphone`, die detailliert im Kapitel 3.5 beschrieben wird

Diese Strategie ermöglicht eine umfassende Bewertung der Algorithmen, indem automatisierte und manuelle Ansätze kombiniert werden, um die Grenzen und Stärken der Methoden aufzuzeigen.

Auch bei der Verwendung des Metaphone-Algorithmus zur Zeichenkettenanalyse wurde beschlossen, die Website **Tilores** nicht zu verwenden, da während der Analyse Fehler in der Kodierung von Wörtern festgestellt wurden.

Ein Fehler bestand darin, dass der Anfangsvokal, der gemäß den Regeln (Die Regeln kann man in Kapitel 2.5 finden.) erhalten bleiben sollte, im Ergebnis nicht angezeigt wurde(siehe Abbildung 19).

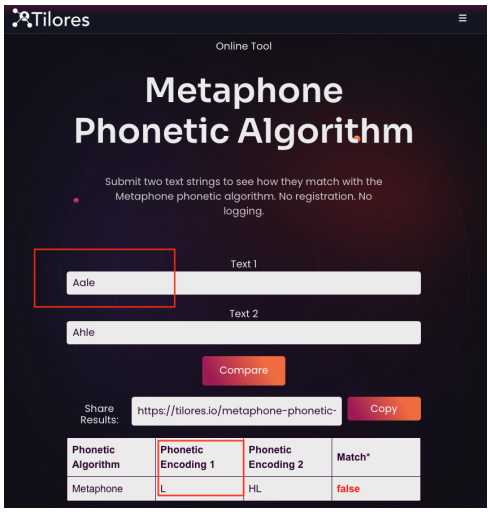

Abbildung 19: Fehler beim Umgang mit Anfangsvokalen in Tilores.

Ebenso sollte der Buchstabe **H** erhalten bleiben, wenn er vor einem Vokal steht und nicht nach den Buchstaben **C**, **G**, **P** gefolgt wird, doch die Ergebnisse der Website ignorierten diese Regel (Die Regeln kann man in Kapitel 2.5 finden. Das Fehler mit "H" kann man in Abbildung 20 sehen).

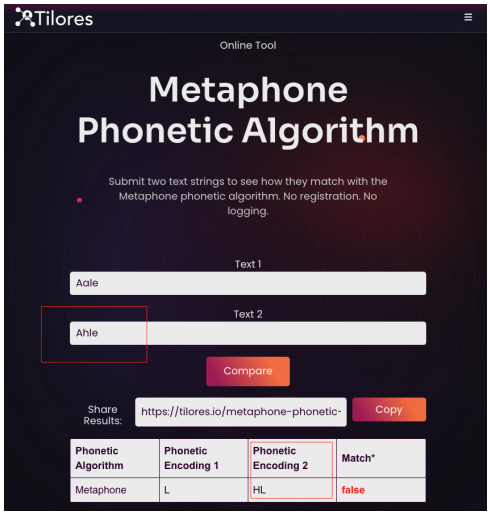

Abbildung 20: Fehler bei der Behandlung des Buchstabens **H** in Tilores.

Als Lösung wurde die Website **Genealogy Online** ausgewählt, die Zeichenketten korrekt nach dem Metaphone-Algorithmus verarbeitet. Der einzige kleine Nachteil besteht darin, dass Wörter einzeln kodiert werden müssen, was jedoch die Gesamteffizienz nicht wesentlich beeinträchtigt.

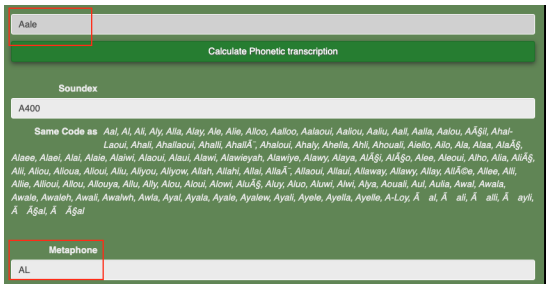

Abbildung 21: Korrekte Kodierung des Wortes *Aale* mit Genealogy Online.

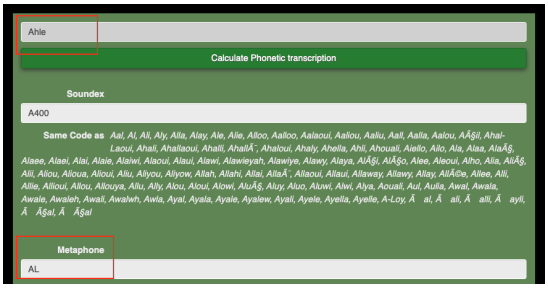

Abbildung 22: Korrekte Kodierung des Wortes *Ahle* mit Genealogy Online.

4.2 Bewertung der Qualität der Ergebnisse von Algorithmen

Um die Funktionsweise verschiedener phonetikbasierter Kodierungsalgorithmen effizient und übersichtlich zu vergleichen, wurde entschieden, die Ergebnisse in einer Tabelle darzustellen. Die Tabelle enthält Spalten mit den zu analysierenden Wörtern sowie Spalten mit den Namen der Algorithmen (Soundex, Metaphone, Kölner Phonetik, Double Metaphone). Jede Zelle zeigt die kodierte Version der Wörter, die durch den entsprechenden Algorithmus generiert wurde, sowie das Ergebnis des Ähnlichkeitsvergleichs (wahr oder falsch).

Die Strategie zur Auswahl der Wörter bestand darin, Wörter mit identischen Buchstaben auszuschließen, um den Einfluss trivialer Übereinstimmungen zu minimieren. Dadurch können die Algorithmen ihre Effektivität und Anpassungsfähigkeit bei der Verarbeitung komplexerer Zeichenketten besser demonstrieren.

Wörter	Soundex	Kölner	Metaphone	Double Metaphone
Aale, Ahle	A400/A400/wahr	05/05/wahr	AL/AL/wahr	AL/AL/wahr
Aar, Ahr	A600/A600/wahr	07/07/wahr	AR/AR/wahr	AR/AR/wahr
Aas, aß	A200/A200/wahr	08/08/wahr	AS/AS/wahr	AS/AS/wahr
Achsel, Axel	A240/A240/wahr	0485/0485/wahr	AXSL/AKSL/falsch	AKSL/AKSL/wahr
Ängste, engste	A523/E523/falsch	06482/06482/wahr	ÄNKST/ENKST/falsch	ANKS/ANKS/wahr
Bay, Bei	B000/B000/wahr	1/1/wahr	B/B/wahr	P/P/wahr
Bällen, bellen	B450/B450/wahr	156/156/wahr	BLN/BLN/wahr	PLN/PLN/wahr
Block, Blog	B420/B420/wahr	154/154/wahr	BLK/BLK/wahr	PLK/PLK/wahr
browsen, brausen	B625/B625/wahr	17386/1786/falsch	BRSN/BRSN/wahr	PRSN/PRSN/wahr
Bug, buk	B200/B200/wahr	14/14/wahr	BK/BK/wahr	PK/PK/wahr
campten, Kempten	C535/K513/falsch	46126/46126/wahr	KMPTN/KMPTN/wahr	KMPT/KMPT/wahr
Celle, Zelle	C400/Z400/falsch	85/85/wahr	SL/SL/wahr	SL/SL/wahr
Krebs, Krepps	K612/K612/wahr	4718/4718/wahr	KRBS/KRPS/falsch	KRPS/KRPS/wahr
Dschinn, Gin	D250/G500/falsch	86/46/falsch	TSXN/JN/falsch	TXN/KN(JN)/falsch
Fähre, faire	F600/F600/wahr	37/37/wahr	FR/FR/wahr	FR/FR/wahr
fällt, Feld	F430/F430/wahr	352/352/wahr	FLT/FLT/wahr	FLT/FLT/wahr
fetter, Vetter	F360/V360/falsch	327/327/wahr	FTR/FTR/wahr	FTR/FTR/wahr
Fiber, Fieber	F160/F160/wahr	317/317/wahr	FBR/FBR/wahr	FPR/FPR/wahr
fließt, fliehst	F423/F423/wahr	3582/3582/wahr	FLST/FLST/wahr	FLST/FLST/wahr
Geld, gellt	G430/G430/wahr	452/452/wahr	JLT/JLT/wahr	KLT(JLT)/KLT(JLT)/ wahr
Graph, Graf	G610/G610/wahr	473/473/wahr	KRF/KRF/wahr	KRF/KRF/wahr
hält, Held	H430/H430/wahr	052/052/wahr	LT/LT/wahr	HLT/HLT/wahr
hängst, Hengst	H523/H523/wahr	06482/06482/wahr	NKST/NKST/wahr	HNKST/HNKST/wahr
Kälten, Kelten	K435/K435/wahr	4526/4526/wahr	KLTN/KLTN/wahr	KLTN/KLTN/wahr
Lachs, Laggs	L200/L200/wahr	548/548/wahr	LXS/LKS/falsch	LKS/LKS/wahr
Laie, Laihe	L000/L000/wahr	5/5/wahr	L/LH/falsch	L/LH/falsch
Leader, Lieder	L360/L360/wahr	527/527/wahr	LTR/LTR/wahr	LTR/LTR/wahr
Leerstelle, Lehrstelle	L623/L623/wahr	57825/57825/wahr	LRSTL/LRSTL/wahr	LRST/LRST/wahr
Mathe, Matte	M300/M300/wahr	62/62/wahr	M0/MT/falsch	M0(MT)/MT/wahr
Miene, Mine	M500/M500/wahr	6/6/wahr	MN/MN/wahr	MN/MN/wahr
Phi, Vieh	P000/V00/falsch	3/3/wahr	F/F/wahr	F/F/wahr
Prinz, Prints	P652/P653/falsch	1768/1768/wahr	PRNS/PRNTS/falsch	PRNS/PRNT/falsch
Raid, Reet	R300/R300/wahr	72/72/wahr	RT/RT/wahr	RT/RT/wahr
Steak, Steg	S320/S320/wahr	824/824/wahr	STK/STK/wahr	STK/STK/wahr
Trend, trennt	T653/T653/wahr	2762/2762/wahr	TRNT/TRNT/wahr	TRNT/TRNT/wahr
Umfeld, umfällt	U514/U514/wahr	06352/06352/wahr	UMFLT/UMFLT/wahr	AMFL/AMFL/wahr
verleasen, Verliesen	V642/V642/wahr	37586/37586/wahr	FRLSN/FRLSN/wahr	FRLS/FRLS/wahr
Wahl, Wal	W400/W400/wahr	35/35/wahr	WL/WL/wahr	AL(FL)/AL(FL)/wahr
Werg, Werk	W620/W620/wahr	374/374/wahr	WKR/WRK/wahr	ARK(FRK)/ARK(FRK)/ wahr
Zuname, Zunahme	Z500/Z500/wahr	86/86/wahr	SNM/SNM/wahr	SNM/SNM/wahr

Abbildung 23: Analyse der Ergebnisse von phonetischen Algorithmen mit positiven und negativen Vergleichen.

Insgesamt wurden 40 Wortpaare analysiert. Die folgenden Ergebnisse zeigen, wie erfolgreich die Algorithmen Soundex, Kölner Phonetik, Metaphone und Double Metaphone die Vergleiche durchgeführt haben:

Algorithmus	**Erfolgreiche Vergleiche**	**Nicht erfolgreiche Vergleiche**	**Erfolgsquote**
Soundex	33	7	82,5%
Kölner Phonetik	38	2	95,0%
Metaphone	32	8	80,0%
Double Metaphone	37	3	92,5%

Analyse aller Algorithmen	**Anzahl der Paare**
Erfolgreich von allen Algorithmen verglichen	27
Nicht erfolgreich von allen Algorithmen verglichen	1

Abbildung 24: Statistik der Ergebnisse von phonetischen Algorithmen beim Vergleich von Wortpaaren.

4.3 Analyse der Vorteile und Nachteile der Algorithmen anhand der Vergleichsergebnisse

Beim Analysieren der Ergebnisse, die in Abbildung 23 dargestellt sind, lassen sich zentrale Einschränkungen des Soundex Algorithmus erkennen. Eine der größten Schwächen ist die strikte Beibehaltung des ersten Buchstabens eines Wortes. Dieses Limit ignoriert die phonetische Relevanz des Anfangsbuchstabens und kann zu fehlerhaften Ergebnissen führen. So werden beispielsweise die Wörter *"Celle"* und *"Zelle"*, trotz ihres identischen Klanges am Anfang, falsch verarbeitet, da "C" und "Z" unterschiedliche Codes erhalten, obwohl sie derselben Gruppe (Code - 2) zugeordnet sind. Eine ähnliche Problematik tritt bei der Wortpaarung *"campten"* und *Kempten"* auf, wo die Buchstaben "C" und "K" denselben Laut haben, aber unterschiedlich kodiert werden.
Da Soundex für die englische Sprache entwickelt wurde, fehlen wichtige phonetische Merkmale wie die Unterstützung für Umlaute (*ä, ö, ü*). Dies führt zu fehlerhaften Vergleichen von Wörtern, die mit Umlauten beginnen, mit phonetisch ähnlichen Wörtern. Ein Beispiel dafür sind die Wörter *"Ängste"* und *"engste"*, die unterschiedliche Codes erhalten, obwohl sie ähnlich klingen.
Darüber hinaus berücksichtigt der Algorithmus keine phonetischen Ähnlichkeiten von Buchstabenkombinationen, was insbesondere in der deutschen Sprache problematisch ist. Dieses Problem zeigt sich beim Vergleich der Wörter *"Phi"* und *"Vieh"*. Obwohl diese Wörter phonetisch eindeutig ähnlich sind, liefern sie ein negatives Ergebnis beim Vergleich, was die Einschränkungen des Algorithmus bei der Verarbeitung solcher Fälle unterstreicht.

Der Kölner Algorithmus erzielte die besten Ergebnisse unter allen getesteten Vergleichsverfahren. Dies ist auf seine Anpassung an die Besonderheiten der deutschen Sprache zurückzuführen, die es ermöglicht, deren phonetische Eigenheiten zu berücksichtigen. So wurden beispielsweise die Wörter *"Prints"* und *"Prinz"*, die von den Algorithmen Soundex, Metaphone und Double Metaphone nicht korrekt verglichen werden konnten, erfolgreich vom Kölner Algorithmus verarbeitet. Der Grund hierfür liegt in der Regel, dass die Buchstabenkombination *"ts"* und der Buchstabe *"z"* denselben Code "8" erhalten.
Ein weiterer Vorteil des Kölner Algorithmus im Vergleich zu Metaphone und Double Metaphone liegt in der korrekten Verarbeitung von Wörtern mit dem Buchstaben *"h"*. Gemäß den Regeln des Algorithmus wird dieser Buchstabe ignoriert, was besonders für die deutsche Sprache von Bedeutung ist, da *"h"* oft nicht ausgesprochen wird. Ein Beispiel hierfür sind die Wörter *"Laie"* und *"Laihe"*: Sowohl der Kölner Algorithmus als auch Soundex konnten diese Wörter korrekt vergleichen. Der Erfolg von Soundex ist ebenfalls auf die Regel zurückzuführen, dass der Buchstabe "h" ignoriert wird.
Zu den fehlerhaften Beispielen zählen die Wörter *"brausen"* und *"browsen"*. Hier liegt der Fehler darin, dass der Buchstabe *"w"* im Kölner Algorithmus einen eigenen Code erhält, während andere Algorithmen diesen Buchstaben ignorieren. Darüber hinaus sind die Wörter *"Dschinn"* und *"Gin"* aufgrund ihrer unterschiedlichen Länge und komplexer phonetischer Kombinationen schwierig korrekt zu kodieren. Alle getesteten Algorithmen lieferten für dieses Wortpaar ein negatives Ergebnis.

Metaphone wurde als eine verbesserte Version von Soundex entwickelt, mit einem wesentlichen Unterschied: der Berücksichtigung von Buchstabenkombinationen und der Zusammenführung von Buchstaben und Kombinationen in gemeinsame Gruppen, die denselben Code erhalten. Beispielsweise erhalten die Kombinationen *"Ph"* und *"V"* im Metaphone denselben Code - *"F"*. Dieses Merkmal zeigt sich deutlich im Beispiel der Wörter *"Phi"* und *"Vieh"*, bei denen Metaphone erfolgreich war, während Soundex ein negatives Ergebnis lieferte.
Allerdings bewahrt der Algorithmus den Anfangsbuchstaben eines Wortes, wenn dieser ein Vokal ist, was zu Fehlern führen kann, insbesondere bei der Arbeit mit der deutschen Sprache. Das Fehlen einer Unterstützung für Umlaute (ä, ö, ü) schränkt die Anwendbarkeit von Metaphone in der deutschen Phonetik ein. Selbst wenn Umlaute in den Algorithmus integriert würden, würden Wörter wie "Ängste" und "engste" unterschiedlich kodiert (*ÄNKST* und *ENKST*) und somit fehlerhafte Ergebnisse liefern.
Trotz des letzten Platzes im Vergleich zu anderen getesteten Algorithmen hat das Metaphone deutliche Vorteile gegenüber Soundex. Diese Vorteile liegen in der flexibleren Verarbeitung von Buchstabenkombinationen und der geringeren Abhängigkeit vom ersten Buchstaben des Wortes. Beispielsweise werden Wortpaare wie "Celle" und "Zelle" oder "campten" und "Kempten" dank der Berücksichtigung phonetischer Ähnlichkeiten von "C", "Z" und "K" korrekt verglichen.

Double Metaphone, eine verbesserte Version des ursprünglichen Metaphone Algorithmus, erzielte Ergebnisse, die fast mit denen der Kölner Phonetik vergleichbar sind. Die Hauptunterschiede zwischen Double Metaphone und Metaphone zeigen sich vor allem in der erweiterten Unterstützung für Fremdsprachen, einschließlich Deutsch. Ein wesentliches Merkmal von Double Metaphone ist die Rückgabe von zwei Codes für ein Wort (einem primären und einem alternativen Code), was die Wahrscheinlichkeit eines erfolgreichen Vergleichs erhöht. So wurden beispielsweise die Wörter *"Mathe"* und *"Matte"*, die von Metaphone aufgrund des Buchstabens "H" falsch verarbeitet wurden, durch die alternative Kodierung von Double Metaphone korrekt verglichen.
Eine weitere Verbesserung ist die Kodierung aller Vokale und des Buchstabens *"W"* am Anfang eines Wortes als *"A"*. Dieses Feature zeigt sich deutlich bei Wörtern wie *"Ängste"* und *"engste"*, bei denen Double Metaphone im Gegensatz zu Metaphone ein positives Ergebnis liefert. Diese Änderung löst das Problem der Kodierung spezifischer Buchstaben wie Umlaute und ermöglicht eine universellere Verarbeitung von Wörtern aus verschiedenen Sprachen.
Darüber hinaus kombiniert Double Metaphone die Buchstaben *"P"* und *"B"* zu einem einzigen Code - "P", was den erfolgreichen Vergleich von Wörtern wie *"Krepps"* und *"Krebs"* ermöglicht. Im ursprünglichen Metaphone Algorithmus werden diese Buchstaben getrennt kodiert, was zu Fehlvergleichen führen kann.
Ein weiterer Vorteil ist die Behandlung der Kombination *"chs"*, die im Double Metaphone als "ks" kodiert wird, während sie im ursprünglichen Metaphone-Algorithmus als *"x"* verarbeitet wird. Beispiele wie *"Axel"* und *"Achsel"* oder *"Lachs"* und *"Laggs"* zeigen, wie Double Metaphone komplexe Wörter genauer vergleicht.
Es gibt jedoch auch Einschränkungen. Double Metaphone hat Schwierigkeiten bei der Verarbeitung von Wörtern mit dem Buchstaben *"H"*. So berücksichtigt der Algorithmus in den Wörtern *"Laihe"* und "Laie" nicht die Besonderheiten der deutschen Sprache, bei denen der Buchstabe *"H"* oft ignoriert wird, wie es die Kölner Phonetik tut. Double Metaphone hingegen verfolgt einen universelleren Ansatz, der zwar die Genauigkeit bei deutschen Wörtern einschränkt, aber die Leistung bei englischen und bei einer Vielzahl von Sprachen Wörtern verbessert.

4.4 Unmöglichkeit des Vergleichs von Zeichenketten in der russischen Sprache

Im Verlauf der Arbeit wurde festgestellt, dass der Einsatz der untersuchten Algorithmen, darunter Soundex, Kölner Phonetik, Metaphone und Double Metaphone, für die russische Sprache nicht geeignet ist. Die Algorithmen sind speziell für die Verarbeitung lateinischer Buchstaben konzipiert, was zu erheblichen Einschränkungen bei der Verwendung für kyrillische Buchstaben führt.

Bei der Kodierung russischer Wörter traten folgende Probleme auf: Die Kodierungsfelder blieben leer, und die Vergleichsergebnisse auf Plattformen wie Tilores waren stets wahr“, unabhängig von der phonetischen Ähnlichkeit der Wörter. Wie in den Abbildungen 25, 26, 27 und 28 zu sehen ist, spiegelt die Kodierung keinen tatsächlichen Algorithmusablauf wider, was die Ergebnisse unbrauchbar macht.

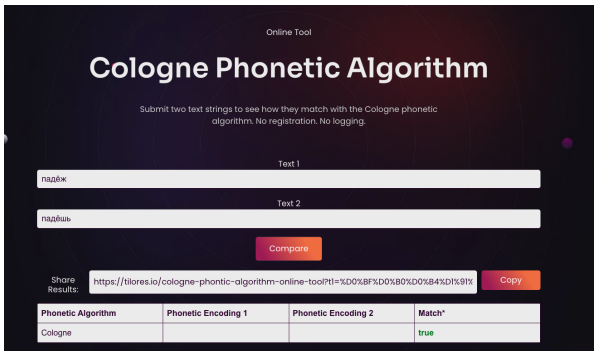

Abbildung 25: Kölner Phonetik: Kodierung auf Russisch ohne Erfolg.

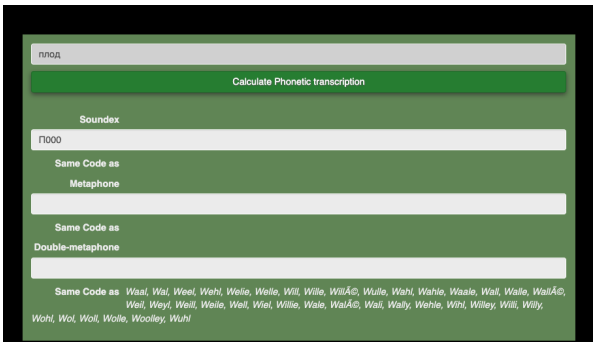

Abbildung 26: Genealogy Online: Ergebnisse für russische Wörter.

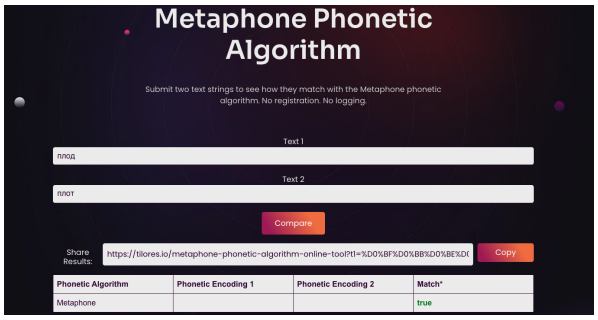

Abbildung 27: Metaphone: Kodierung auf Russisch ohne Erfolg.

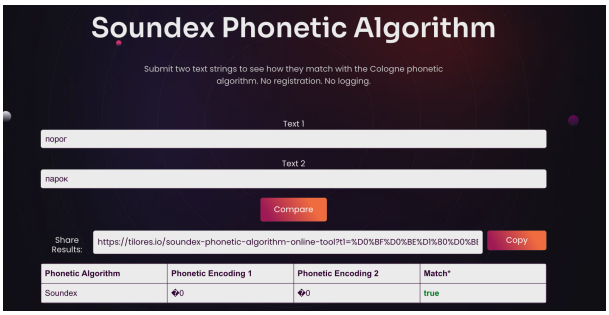

Abbildung 28: Soundex: Ergebnisse für russische Wörter.

Abbildung 29: Versuch der Kodierung russischer Wörter mit verschiedenen Algorithmen.

Dieses Verhalten zeigt, dass die zugrunde liegenden Implementierungen der Algorithmen keine Unterstützung für kyrillische Schrift bieten. Es bleibt festzuhalten, dass die Unmöglichkeit des Vergleichs russischer Wörter nicht als eine einfache technische Einschränkung interpretiert werden kann, sondern als eine fundamentale Eigenschaft der Algorithmen, die für Sprachen mit lateinischen Buchstaben entwickelt wurden.

5 Schlussfolgerung

5.1 Schlussfolgerungen zur effektiven Nutzung der Algorithmen

In diesem Abschnitt werden die Ergebnisse der vergleichenden Analyse der Algorithmen Soundex, Kölner Phonetik, Metaphone und Double Metaphone zusammengefasst. Zudem werden ihre Stärken und Schwächen in Abhängigkeit von den jeweiligen Aufgaben erörtert. Wie ausführlich in Kapitel 4.3 beschrieben und durch die Statistik in Abbildung 24 bestätigt, sollte die Auswahl eines Algorithmus auf den sprachlichen Besonderheiten und den phonologischen Eigenschaften der zu vergleichenden Zeichenketten basieren.

Wie bereits erwähnt, sollten diese Algorithmen nicht für den Vergleich von Zeichenketten in Russisch verwendet werden.

Der Algorithmus **Soundex** wurde ursprünglich für die englische Sprache entwickelt. Seine strikte Regel, den ersten Buchstaben eines Wortes beizubehalten, sowie die fehlende Berücksichtigung von Buchstabenkombinationen machen ihn für deutsche Wörter weniger geeignet. Trotzdem kann Soundex in Aufgaben nützlich sein, bei denen Zeichenketten mit demselben Anfangsbuchstaben verglichen werden. Im Rahmen dieser Untersuchung zeigte der Algorithmus eine Erfolgsquote von 82,5%, was trotz seiner Einschränkungen als akzeptabel angesehen werden kann.

Der **Kölner Algorithmus** erzielte mit 95% die besten Ergebnisse unter allen getesteten Methoden. Diese Leistung ist auf seine Anpassung an die deutsche Sprache zurückzuführen, die es ermöglicht, spezifische phonologische Eigenschaften wie denselben Code für Buchstabenkombinationen *"ts"* und *"z"* zu berücksichtigen. Dennoch können Fehler bei der Verarbeitung englischer Wörter auftreten, beispielsweise bei der Buchstabe *"W"*, die im Englischen oft nicht ausgesprochen wird, aber im Deutschen anders kodiert wird.

Metaphone unterscheidet sich von Soundex durch die Berücksichtigung von Buchstabenkombinationen, was seine Genauigkeit erhöht. Dennoch ist der Algorithmus anfällig für Fehler bei der Verarbeitung von Wörtern, die mit unterschiedlichen Vokalen beginnen, und ordnet die Buchstaben *"P"* und *"B"* nicht derselben phonologischen Gruppe zu. Die Einschränkungen des Algorithmus sind hauptsächlich auf seine Ausrichtung auf die englische Sprache zurückzuführen, was seine Anwendbarkeit auf deutsche Wörter einschränkt. Trotzdem erzielte Metaphone 80% erfolgreiche Vergleiche, was ein zufriedenstellendes Ergebnis ist.

Double Metaphone, eine erweiterte Version von Metaphone, zeigte Ergebnisse, die mit der Kölner Phonetik vergleichbar sind, und erreichte eine Erfolgsquote von 92,5%. Seine Stärke liegt in der Unterstützung mehrerer Sprachen, was ihn für internationale Wörter geeignet macht. Die doppelte Kodierung, die sowohl einen primären als auch einen alternativen Code bietet, erhöht die Wahrscheinlichkeit eines erfolgreichen Vergleichs. Dennoch kann der Algorithmus bei der Verarbeitung spezifischer deutscher Buchstabenkombinationen wie *"ts"* und *"z"* auf Einschränkungen stoßen.

Daher sollte die Wahl des Algorithmus auf den spezifischen Anforderungen der Aufgaben und den Eigenschaften der zu vergleichenden Zeichenketten basieren. Für die deutsche Sprache sind Kölner Phonetik und Double Metaphone die besten Optionen, wobei letzterer aufgrund seiner angegebenen Universalität potenziell auch mit mehrsprachigen Wörter umgehen kann. Es ist jedoch zu beachten, dass diese Annahme im Rahmen dieser Studie nicht überprüft wurde. Soundex und Metaphone, die bestimmte Einschränkungen aufweisen, eignen sich besser für die Analyse von Zeichenketten in der englischen Sprache.

5.2 Bewertung der Zielerreichung

Die in dieser Arbeit zu Beginn formulierten Ziele, einen umfassenden Überblick über Software und Algorithmen zur phonetischen Ähnlichkeitsanalyse von Zeichenketten zu erstellen, wurden durch die vorherigen Kapitel erfolgreich erreicht.

Anhand einer Tabelle mit Homophonen konnte demonstriert werden, wie präzise die einzelnen Algorithmen für die deutsche Sprache bei der phonetischen Ähnlichkeitsanalyse von Zeichenketten arbeiten. Dabei wurde nicht nur ermittelt, welcher Algorithmus in welchem Kontext am besten geeignet ist, sondern auch die Vor- und Nachteile jedes Algorithmus detailliert herausgearbeitet.

Im Verlauf der Untersuchung wurde klar, dass nicht jedes Werkzeug zuverlässig Wörter kodiert oder genaue Ergebnisse liefert. Daher ist es von zentraler Bedeutung, den passenden Algorithmus oder das passende Werkzeug je nach Anwendungsfall sorgfältig auszuwählen. Um dies zu verdeutlichen, wurden in dieser Arbeit verschiedene Werkzeuge – sowohl Webanwendungen als auch Python-Bibliotheken – miteinander verglichen. Durch die Analyse der Kodierungsregeln der Algorithmen konnte festgestellt werden, welche Werkzeuge bei der phonetischen Kodierung besonders präzise sind.

Darüber hinaus wurde im Rahmen dieser Arbeit ein einfaches Python-Programm entwickelt, das auf Basis vorhandener Bibliotheken eine Analyse von Zeichenketten nach phonetischer Ähnlichkeit ermöglicht. Dieses Programm bot eine praxisorientierte Unterstützung bei der Bewertung der Algorithmen und lieferte wertvolle Einblicke in deren Funktionsweise.

6 Selbstständigkeitserklärung

Hiermit erkläre ich, dass diese Arbeit von mir selbstständig und ohne fremde Hilfe angefertigt wurde. Ich habe Übersetzungstools wie Yandex und DeepL genutzt, um mir unbekannte Wörter ins Deutsche zu übersetzen. Der Einsatz von ChatGPT beschränkte sich ausschließlich auf die Überprüfung der grammatikalischen Korrektheit von Sätzen und kleinere Verbesserungen des Textes im wissenschaftlichen Stil. Insbesondere versichere ich, dass ich mich mit allen wörtlichen und inhaltlichen Übernahmen aus anderen Quellen vertraut gemacht habe und diese in meiner Arbeit entsprechend gekennzeichnet sind.

Ernaz Erkinbekov, 29.11.2024

7 Quellen

7.1 Literatur

[1] Ferrand, L., & Grainger, J.: *Homophone interference effects in visual word recognition*, (2003) In: Quarterly Journal of Experimental Psychology, 56(3), 403–419.

[2] Frankie Patman & Leonard Shaefer: *Is Soundex Good Enough for You? On the Hidden Risks of Soundex-Based Name Searching*, (Januar 2003). 20 Seiten.

[3] Christopher Deringer: *Analyse politischer Meinungen und Erkennung von Stilfiguren*, (Februar, 2024). Technische Universität Wien

[4] Postel Hans Joachim: *Die Kölner Phonetik. Ein Verfahren zur Identifizierung von Personennamen auf der Grundlage der Gestaltanalyse.*, (1969). In: IBM-Nachrichten. Seite 925.

[5] Binstock, A., & Rex, J.: *Practical Algorithms for Programmers.* Addison-Wesley.,(1995). 577 Seiten.

[6] Alexander Beider & Stephen P. Morse: *"Phonetic Matching: A Better Soundex"*. Erschienen im *Association of Professional Genealogists Quarterly*, (März 2010).

7.2 Verwendete Werkzeuge

Im Rahmen des wissenschaftlichen Forschungsprojekts wurden verschiedene frei verfügbare Ressourcen und Anwendungen genutzt, um Algorithmen zur phonetischen Ähnlichkeit zu testen und zu vergleichen. Nachfolgend sind die wichtigsten Ressourcen und ihre Verwendungszwecke aufgeführt:

- **Soundex von Tilores:** Ein Online-Tool zur phonetischen Analyse mithilfe des Soundex-Algorithmus.
  URL: `https://tilores.io/soundex-phonetic-algorithm-online-tool`
- **Kölner Algorithmus von Tilores:** Ein Online-Tool, das den Kölner Phonetischen Algorithmus implementiert.
  URL: `https://tilores.io/cologne-phontic-algorithm-online-tool`
- **Metaphone von Tilores:** Eine weitere Online-Anwendung zur phonetischen Analyse mit dem Metaphone-Algorithmus.
  URL: `https://tilores.io/metaphone-phonetic-algorithm-online-tool`
- **Genealogy Online:** Eine Plattform zur genealogischen Forschung, die phonetische Algorithmen wie Soundex und Metaphone anbietet.
  URL: `https://www.genealogieonline.nl/en/naslag/phonetic/`
- **Phonetisch-Bibliothek:** Eine Python-Bibliothek, die den Soundex Algorithmus implementiert.
  URL: `https://pypi.org/project/phonetisch/`
- **Kölner Bibliothek:** Eine Python-Bibliothek, die den Kölner Phonetischen Algorithmus implementiert.
  URL: `https://pypi.org/project/cologne-phonetics`
- **Metaphone-Bibliothek:** Eine Python-Bibliothek für die Arbeit mit dem Metaphone-Algorithmus.
  URL: `https://pypi.org/project/Metaphone/`
- **Virtual Enviroment:** Virtuelle Python-Umgebungen wurden verwendet, um die getesteten Bibliotheken zu isolieren und sicherzustellen, dass Abhängigkeiten konsistent bleiben.
  Anleitung zur Einrichtung: `https://packaging.python.org/en/latest/guides/installing-using-pip-and-v`
  `#create-and-use-virtual-environments`
- **Liste deutscher Homophone:** Eine umfangreiche Sammlung deutscher Homophone, die für Tests der phonetischen Algorithmen genutzt wurde.
  URL: `https://de.wiktionary.org/wiki/Verzeichnis:Deutsch/Homophone`