



UNIVERSITÄT LEIPZIG
FAKULTÄT FÜR MATHEMATIK UND INFORMATIK
INSTITUT FÜR INFORMATIK

**Verfahren zur Datenintegration
strukturell heterogener Quellen
im Kontext der eHumanities**

Bachelorarbeit

Leipzig, September 2010

vorgelegt von
Pansch, David
Studiengang Informatik

**Betreuender Hochschullehrer: Prof. Dr. Gerhard Heyer
Abteilung Automatische Sprachverarbeitung**

Inhaltsverzeichnis

1	Einleitung	3
2	Genutzte Daten	6
2.1	Herkunft	6
2.2	Verarbeitung	8
3	Methoden	11
3.1	Einordnung der Ansätze	11
3.2	Verwendete Verfahren	13
3.2.1	Name Similarity	14
3.2.2	Path Depth Similarity	15
3.2.3	Cosine Similarity	15
3.2.4	Dice Similarity	16
3.2.5	Absolute Frequency Similarity	17
3.2.6	Relative Frequency Similarity	17
3.2.7	Content Type Similarity	17
3.3	Ausgabe	18
4	Evaluation	20
4.1	Ungewichtet	22
4.2	Wahl der Gewichte	26
4.3	Gewichtet	29
4.4	Wertung	31
4.5	Weitere Parameter	32
5	Lessons Learnt und Further Work	35
6	Zusammenfassung	38
7	Literaturverzeichnis	39
8	Anlagen	41
9	Erklärung	48

Abstract

Informationen haben oft verschiedene Quellen und Formate, ungeachtet ihres Inhaltes. Eine Vereinheitlichung der Strukturen dieser Daten zur Überführung in Datenbanken zum Zwecke der Speicherung und Auswertung ist dann nicht ohne Weiteres möglich. Auch digitalisierte Textinformationen können in unterschiedlichen Formaten vorliegen. Für eine Datenintegration, insbesondere die Überführung von XML-Inhalten in ein bestehendes Datenbankschema, soll ein Schema-Mapping durchgeführt werden. Verschiedene Verfahren werden implementiert und ihre Funktionsweise auf Brauchbarkeit bei unterschiedlichen Input-Formaten und -Kombinationen untersucht.

1 Einleitung

Die umfangreiche Speicherung und Auswertung von unterschiedlichsten Daten ist heutzutage keine Seltenheit mehr. In eAQUA (siehe [22]) werden verschiedenste Daten und Metadaten gehalten, welche zum einen als Grundlage für die Erkenntnisgewinnung genutzt werden und zum anderen die Ergebnisse der Text-Mining-Prozesse mit Informationen anreichern.

Das Projekt eAQUA vereint moderne Text-Mining-Verfahren mit antiken Texten und liegt damit klar im Bereich der eHumanities. Die eHumanities beschreiben im Gegensatz zu den Digital Humanities, welche sich auf die alleinige Digitalisierung der altertümlichen Texte beschränken, die automatische Wissensgeneration auf den zahlreichen Korpora durch komplexe Algorithmen. In eAQUA arbeiten Altertumswissenschaftler mit Informatikern interdisziplinär zusammen. Die Zielsetzung ist ein Web-Portal, in dem eben dieses generierte Wissen zum Zwecke der Forschung zur Verfügung gestellt wird.

Doch wenn die Informationen in stark heterogenen Formen und Formaten vorliegen, ist eine triviale und einheitliche Überführung in die Datenbanken nicht ohne Weiteres möglich. Insbesondere ist es schwierig, automatisiert Redundanzen in den Daten zu finden und ihre Strukturen aufeinander abzubilden. Da dies in händischer Durchführung ebenfalls mit einem großen Aufwand verbunden ist, ist das Ziel dieser Arbeit die Entwicklung eines Programms, welches Gemeinsamkeiten in verschiedenen und heterogenen Daten erkennt und die Vergleichsergebnisse graphisch und tabellarisch ausgibt. Dieses Programm wird mit dem Arbeitstitel „Geminus“¹ (siehe [23]) im Rahmen von eAQUA entwickelt.

Die wesentliche Aufgabenstellung ist der Vergleich von Datenbanken mit XML-Dateien, mit dem Hintergrund, die Inhalte der XML-Dateien in bereits bestehende Datenbankschemata einzufügen. Hierbei liegt das hauptsächliche Augenmerk auf Textdaten und in den Korpora enthaltenen Metadaten und Annotationen.

Als Metadaten werden hier nicht etwa hintergründige, strukturelle Informationen betra-

¹Lateinisch: Zwilling.

chtet, sondern zusätzliche Fakten zum Text, etwa der Autor, das Entstehungsdatum oder der Entstehungsort. Desweiteren werden innerhalb dieser Arbeit die Begriffe ‘Daten’ und ‘Informationen’ synonym behandelt. Es wird hier nicht durch Kategorien wie Struktur oder semantischer Wertung unterschieden.

Das folgende Beispiel soll die Problematik verdeutlichen: Steht in einer Datenbank „Korpus“ in der Tabelle „Sätze“ in der Spalte „Satz“ eine Zeile mit dem Inhalt

καὶ περὶ τὴν κώμην ἱερὸν Σαράπιδος Ὀσορομνή

und in einem XML-Dokument an einer Stelle in Betacode²

```
<TEI.2><text><body><div1>
...
<p>
    kai\ peri\ th\n kw/mhn
</p>
<p>
    i(ero\n *sara/pidos *)osoromnh/
</p>
...
</div1></body></text></TEI.2>
```

so sollte in diesem Minimalbeispiel automatisch herausgestellt werden, dass seitens der Datenbank unter *Korpus/Sätze/Satz* inhaltlich das Gleiche steht, wie im XML-Dokument in *TEI.2/text/body/div1/p*.

Obwohl das Programm spezifisch für das eHumanities-Projekt eAQUA entwickelt wird, soll es möglichst generisch bleiben und auf einer breiten Masse verschiedener Daten einsetzbar sein. Um das zu erreichen, werden unterschiedliche Verfahren zur Datenintegration prototypisch ausprobiert und die Ergebnisse analysiert. Diese Untersuchung wird folglich keine Tiefenarbeit. Es werden günstige Methoden gesucht und verwendet, allerdings ist das Ziel nicht die Optimierung dieser für spezifische Datengrundlagen. Damit ein möglichst breites Spektrum an Einsatzgebieten abgedeckt werden kann, mit dem Anspruch, nicht nur im Bereich der eHumanities nützlich zu sein, und gute Gewichte für Vergleiche von Datenbanken untereinander, XML-Dokumenten untereinander oder Datenbanken mit XML-Dokumenten zu finden, ist die Dimension der Arbeit die Breite.

Weiterhin versteht man unter Datenintegration oder der Homogenisierung von Daten häufig Prozesse mit dem automatischen Zusammenführen oder ‘Merging’ der Daten am Ende, wie in [11]. In dieser Arbeit wird es nicht darum gehen, eine neue Ausgabe zu erzeugen, die eine Vereinheitlichung der Eingabeinformationen ist. Es soll ausschließlich ein Mapping geben, welches ähnliche Strukturelemente mit einer gewissen Sicherheit aufeinander abbildet.

In den folgenden Abschnitten wird zunächst auf die Daten eingegangen, die zum Testen und Evaluieren benutzt werden. Sie enthalten hauptsächlich griechische Texte, sowohl in

²Die Darstellung griechischer Buchstaben mitsamt ihrer diakritischen Zeichen durch lateinische Buchstaben mit nachgestellten Klammern, Schrägstrichen und anderen Zeichen.

Betacode als auch in Unicode. Zur Abgrenzung sind allerdings auch englische Korpora darunter. Daraufhin werden dann die einzelnen Methoden im Detail vorgestellt und anschließend werden die Ergebnisse ausführlich evaluiert. Nachdem auf die Erfahrungen eingegangen wird und mögliche fortführende Arbeiten genannt werden, wird die Arbeit abschließend zusammengefasst.

2 Genutzte Daten

Das Programm Geminus erwartet stets zwei verschiedene Eingabekorpora, im Folgenden als *Inputs* bezeichnet. Eine Datenbank mit einer oder mehrerer spezifizierbaren Tabellen kann als Input dienen, oder aber auch ein Ordner, welcher XML-Dokumente enthält.

An dieser Stelle wird auf die Herkunft und Inhalte der verwendeten Dokumente eingegangen, bevor deutlich gemacht wird, wie diese Daten im Speziellen eingelesen und verarbeitet werden.

2.1 Herkunft

Im Rahmen der Tests und der Evaluation von Geminus wurden als Eingabedaten Inhalte aus Datenbanken und XML-Dokumenten gewählt. Die Datenbanken waren projektintern bereits vorhanden.

Für den Vergleich herangezogen werden folgende Tabellen mit ihren Spalten:

- authors
 - a_id (projektinterne Autornummer)
 - corpora_author_id (originale Autornummer)
 - author (Name)
 - dating (Jahresangaben)
 - epithets (Klassifikation, e.g. Genre)
 - female (Geschlecht)
 - geography (Ort)
- works
 - work_id (projektinterne Werknummer)
 - corpora_author_id
 - corpora_work_id (originale Werknummer)
 - work (Name)
- sentences
 - s_id (Satznummer)
 - sentence (bereits segmentierter Satz)

Die verwendeten Korpora unterliegen diesem Schema, was allerdings nicht bedeutet, dass sämtliche Informationen auch vorliegen. Oft sind Einträge wie „epithets“, „dating“ und sogar „author“ leer oder ‘NULL’.

Das Perseus-Projekt (siehe [24]) stellt einen Großteil der XML-Dokumente zur Verfügung. Neben der *Duke Databank of Documentary Papyri*, kurz DDbDP, werden zur

Abgrenzung auch englischsprachige Dokumente aus dem Angebot von Perseus verwendet, und zwar Texte zur amerikanischen Geschichte des 19. Jahrhunderts und von den Richmond-Times-Dispatch-Korpora (siehe [26]).

Bei der DDbDP handelt es sich um eine Sammlung von mehreren tausend Dokumenten, die Inschriften von Papyri, Tonscherben und Holztafeln in digitalisierter Form enthalten. Sie ist zu rund 94,5% Griechisch. Der zu vernachlässigende Rest ist hauptsächlich Englisch und Latein.

Weitere XML-Dateien kamen vom King's College London (siehe [21]). Deren Epiduke-Dokumente unterscheiden sich in wesentlichen Punkten von der DDbDP, die Perseus anbietet, obwohl sie inhaltlich annähernd identisch sind.

Zum einen beinhalten die Perseus-Dokumente Betacode, wohingegen die Epiduke-Dateien in Unicode vorliegen. Zusätzlich sind die Inhalte von Perseus in wesentlich weniger Dateien aufgeteilt. Während die DDbDP von Perseus aus 256 Dateien (82,1 MiB) besteht, umfasst die des King's Colleges ganze 54776 einzelne Dokumente (252,3 MiB), jede XML-Datei mit einem eigenen Header. Trotzdem sind die eigentlichen textuellen Inhalte, wie bereits angedeutet, gleich.

Alle verwendeten Dokumente haben gemein, dass sie mit dem TEI-Standard (siehe [25]) enkodiert sind. Das TEI-Konsortium entwickelt Standards und Richtlinien zur Repräsentation von digitalisierten Texten im XML-Format. Dabei werden trotzdem genug Freiheiten für Heterogenität in den Daten gelassen, nicht nur weil die Entwicklung der XML-Schemata ein Prozess ist und gewisse Repräsentationen somit veralten können.

In den standardisierten XML-Dateien finden sich folgende Tags neben zahlreichen weiteren:³

- title
- placeName
- date
- idno (identifying number)
- text

Andere Tags geben Aufschluss über Verläge, verwendete Sprachen, Änderungen am Dokument, Bibliotheken oder aber direkt textbezogene Informationen wie Lücken, Ergänzungen oder einfach nur Absätze. Die oben genannten Elemente aber sind die, die möglicherweise ein Pendant in dem Datenbankschema finden. Objektiv sind beispielsweise „geography“ und „placeName“, sowie „sentences“ und „text“ gleich.

Nachfolgend wird erläutert, wie die strukturellen und textuellen Informationen eingelesen, verarbeitet und schließlich miteinander verglichen werden.

³Auf die Hierarchie und die Tiefenebene der Elemente wird an dieser Stelle nicht eingegangen.

2.2 Verarbeitung

Die zentrale Struktur in Geminus, die später dem Vergleich unterzogen wird, ist ein *Path*. Für Datenbanken ist ein Path verhältnismäßig einfach. Sein vollständiger Bezeichner besteht aus einer slash-separierten Kombination aus Datenbankname, Tabellename und Spaltenname, also zum Beispiel „Korpus/Sätze/Satz“. Der jeweilige abstrakte Inhalt des Paths ist alles, was in der Datenbank in dieser Spalte steht.

Bei XML-Dokumenten ist ein Path ein (Element-)Knoten im DOM-Baum (also insbesondere keine Kommentarknoten, Textknoten oder Attributknoten).⁴ Der Bezeichner eines Paths aus einem XML-Dokument ist angelehnt an die XPath-Notation, so dass jede Ebene ein (slash-separiertes) Segment darstellt. In einer Dokumentenkollektion werden in einem Path die Inhalte der einzelnen gleichen Paths der einzelnen Dateien kumuliert. Paths sind somit unabhängig von der Anzahl der Dokumente, aus welchen ihr Inhalt stammt.

Ein Path enthält sowohl seine eigenen Textknoten als auch die Textknoten aller seiner Kindknoten. Folglich ist beispielsweise der abstrakte Inhalt des Wurzelknotens die gesamte Dokumentenkollektion. Der Hintergrund zu dieser Entscheidung liegt darin, dass die Text-Inhalte in den Datenbanken oft schon fertige, kaum bis gar nicht annotierte Texte sind, während Textknoten in den XML-Dateien oft Wörter oder ganze Abschnitte in Unterknoten haben, versehen mit Tags wie `<unclear>`, `<line>` oder auch `<number>`. Würden deren Inhalte nicht Bestandteil des Elternknotens (= des hauptsächlichen Textknotens) sein, würden sie beim Vergleich fehlen. Durch das Hinzuziehen der Kindknoten entstehen wesentlich Redundanzen. Je tiefer sich ein Wort in der XML-Struktur befindet, in desto mehr Paths ist es zu finden. Dies wirkt sich allerdings nicht nachteilig aus.

In Abhängigkeit diverser Parameter werden die eingelesenen Zeilen noch verarbeitet, bevor die Informationen, die die einzelnen Vergleichsmethoden benötigen, im Arbeitsspeicher abgelegt werden.

Betacode Konversion: Diese Option – speziell für griechische Texte – ermöglicht die Konversion von UTF-8-Griechisch nach Betacode, falls einer der Inputs in Betacode ist und der andere nicht. Ohne diese Möglichkeit kann in einem solchen Fall kein Stringvergleich funktionieren. Die Konversion von Betacode nach UTF-8 ist zwar ebenso denkbar, allerdings nicht implementiert, da sonst unkontrolliert auch aus Wörtern in lateinischen Buchstaben ungewollt Griechisch würde, wie zum Beispiel $\lambda\alpha\nu\gamma\upsilon\alpha\gamma\epsilon$ (ursprünglich *language*).

Tokenisierung: Die Interpunktion der ursprünglichen Zeilen ist für kein verwendetes Verfahren von Interesse. Lediglich die von Paths beinhalteten Wörter werden verarbeitet. Dazu werden sämtliche Satzzeichen entfernt, da diese hier nicht als Wörter gelten sollen.

⁴Für XML-Dokumente kann im Zuge des Einlesens entschieden werden, ob eventuell angegebene DTDs ausgewertet werden sollen oder nicht. Für die Evaluation wurde eine XML-Validierung grundsätzlich unterdrückt, da manche Dateien eine lokale DTD angegeben hatten, die nicht existierte.

Normalisierung: Dieser Begriff meint hier eine progressive Zeichenersetzung, basierend auf einer Buchstabenliste. Bei griechischen Zeichen können beispielsweise alle komplexen, mit diakritischen Zeichen besetzten Buchstaben durch ihre Grundform ersetzt werden, so dass aus $\acute{\alpha}$, $\tilde{\alpha}$, A oder $\tilde{\alpha}$ jeweils ein einfaches α wird. Diese Buchstabenliste lässt sich sehr einfach auch für Betacode anpassen und einsetzen. Die Normalisierung dient dazu, Wörter im Stringvergleich als identisch anzuerkennen, die sich nur in Akzenten oder Spiritus unterscheiden. Außerdem lässt sich mit dieser Methode eine uneinheitliche Kodierung des Griechischen umgehen, falls zum Beispiel ein Input „combined diacritics“ (2 Zeichen für $\tilde{\alpha}$) benutzt und der andere nicht (1 Zeichen für $\tilde{\alpha}$), oder ein Input verwendet ϕ und im anderen steht φ für den Buchstaben *phi*.

Ignorieren von Nummern: Dies kann sinnvoll sein, wenn in einem Input etwas steht wie $\langle \text{num} \rangle \beta \langle / \text{num} \rangle$ und in dem anderen $\langle \text{num} \rangle 2 \langle / \text{num} \rangle$. Enthalten solche Inputs viele Zahlen, weil es zum Beispiel Papyri aus dem Finanzwesen sind, entstehen so viele Wörter, die einer der Inputs enthält und der andere nicht, was das Ergebnis verschlechtern kann. Eine andere Sichtweise auf diese Funktion ist die Eindämmung der Komplexität, da die Menge der Wörter abnimmt und Zahlen in manchen Kontexten nicht unbedingt Gegenstand eines inhaltlichen Vergleichs sein müssen.

Zusätzlich gibt es Parameter, die es ermöglichen, Wörter und Paths im Nachhinein zu streichen.

Nachdem alle Wörter ausgezählt wurden, lassen sich die prozentual häufigsten Wörter (Wert konfigurierbar) und die Wörter mit Frequenz 1 entfernen.

Für XML-Inputs gibt es außerdem die Möglichkeit, Paths zu entfernen, die nur in (konfigurierbaren) Teilmengen der Dateien tatsächlich vorkommen. Diese Funktion war ursprünglich ausschließlich zur Komplexitätsreduktion gedacht, da die Anzahl der durchzuführenden Vergleiche und auch die der im Arbeitsspeicher gehaltenen Ergebnisse das Produkt der Menge der Paths aus den verschiedenen Inputs ist. Im Nachhinein erwies sich diese Funktion allerdings als sehr sinnvoll. Falls ein Path zum Beispiel nur in 10% der Dokumente auftritt, ist dieser Path und ein Mapping auf den anderen Input oft uninteressant, da er durch seine Seltenheit kein wesentlicher Bestandteil dieses Inputs ist. Sollte es anders sein, lässt sich das Programm entsprechend konfigurieren.

Für die Evaluation wurde der Wert 0,5 gewählt. Das heißt, dass ein Path nur im Output erscheint, wenn er mindestens in der Hälfte aller Dokumente auftritt.

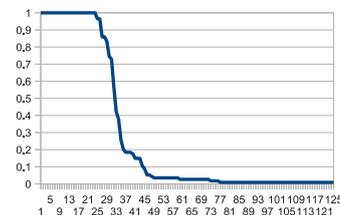


Abbildung 1: In *o.camb* kommen viele Paths in weniger als der Hälfte der Dokumente vor.

In den Abbildungen 1 und 2 ist die Verteilung der Paths auf die Dokumente in Epiduke zu sehen. *O.camb* ist hierbei ein Teil der DDbDP. Die Abszisse zeigt die Paths und auf der Ordinate ist der jeweilige Wert abgetragen, in wie vielen Dokumenten dieser Path vorkommt. Für die mehr als 50'000 Epiduke-Dokumente wird sehr deutlich, dass bei weitem nicht jeder Path, der in den einzelnen Dateien auftritt, für den Vergleich relevant ist. Ein Großteil der Paths ist extrem selten. Der Filter wird also eingesetzt, um die Mappings auf die Paths zu beschränken, aus denen die Dokumente im wesentlichen bestehen, was außerdem die Anzahl der notwendigen Vergleiche erheblich reduziert.

Nachdem nun die Datengrundlagen erläutert wurden, beleuchtet der folgende Abschnitt ausführlich die verwendeten Vergleichsmethoden.

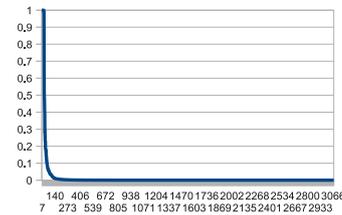


Abbildung 2: Die meisten Paths in Epiduke sind in extrem wenigen Dokumenten vertreten.

3 Methoden

Die Idee des Schema-Matchings zur Datenintegration ist keine neue, allerdings zielen bisherige Arbeiten zu diesem Thema ([3, 14, 18, 16, 1]) meist nicht auf die Zusammenführung von Textdaten ab, obgleich hier auch Übereinstimmungen in zusätzlichen Angaben wie Jahreszahlen oder Ortsnamen gefunden werden sollen. Insbesondere für den Vergleich griechischer Korpora, die sowohl in Datenbanken als auch im XML-Format vorliegen, lag eine Neuentwicklung nahe, da die Anforderungen (Umgang mit Beta-code, spezifische Inputs, Experimentieren mit verschiedenen Verfahren) so am einfachsten umgesetzt werden konnten.

In diesem Abschnitt werden vorhandene Ansätze kategorisiert und beschrieben. Anschließend wird die Funktionsweise der letztendlich implementierten Verfahren geschildert.

3.1 Einordnung der Ansätze

Die Taxonomie der Schema-Matching-Ansätze ist verhältnismäßig umfangreich. Einzelne Aspekte lassen sich fast beliebig tief detaillieren, hier soll an dieser Stelle aber nur eine Übersicht gegeben werden.

Eine Möglichkeit der Kategorisierung von Algorithmen sind die *Dimensionen* des Verfahrens, wie sie auch in [18] beschrieben sind.

Input: Die Eingabedimensionen beschreiben, auf welchen Typen von Informationen die Algorithmen arbeiten. Im Allgemeinen sind damit die einlesbaren Dateitypen gemeint, spezifischer die konzeptuellen Modelle, in welchen Ontologien oder Schemata dargestellt werden (wie RDF oder OWL, beziehungsweise OO oder ER Modelle, aber auch XML). Ein weiterer Aspekt ist der konkrete Umgang mit den vorhandenen Daten und die Verarbeitung der strukturellen und inhaltlichen Informationen. Darauf wird im Folgenden ausführlicher eingegangen.

Prozess: An dieser Stelle wird sich nicht auf die formale Klassifizierung von Algorithmen bezogen, sondern auf den Einfluss der Berechnungsweise auf die Exaktheit der Ergebnisse. In diesem Kontext gibt es die Unterteilung in ‘näherungsweise’ und ‘exakt’. Exakte Algorithmen berechnen die absolute Lösung zu einem Problem, während im anderen Fall zugunsten der Performanz ein Teil der Exaktheit geopfert wird.

Eine weitere Prozessdimension stellt die Interpretation der Eingabe dar. Es ergeben sich drei Klassen, je nachdem, ob der Input rein syntaktisch verarbeitet wird, ob es eine Anwendung semantischer Theorien gibt oder ob externe Ressourcen benutzt werden.

Output: Am Ende des Prozesses steht eine Zuordnung der Elemente und Strukturen zwischen den Eingabedaten. Im Resultat lässt sich zum einen unterscheiden, ob es ein Sicherheitsmaß für ein Mapping gibt oder ob es einfach eine Zuordnung definitiv gibt oder nicht. Zum anderen lassen sich Relationen zwischen Elementen

auch ausführlicher ausdrücken, als durch Gleichheit ($=$), und zwar zum Beispiel auch mit Subsumtion (\sqsubseteq) oder Inkompatibilität (\perp).

Allgemein lassen sich die Methoden durch folgende Unterteilungen *klassifizieren*, siehe auch [14, 16].

schemenbasiert vs. instanzbasiert: Die schemenbasierten Verfahren betrachten ausschließlich die Schema-Informationen, wie Namen, Beschreibungen, Relationen oder Beschränkungen. Instanz- bzw. inhaltsbasierte Ansätze betrachten die Inhalte und erstellen zum Beispiel Statistiken für den Vergleich, wie in [13]. Schemenbasierte Verfahren eignen sich demnach, Dokumente zu vergleichen, die unabhängig vom Inhalt einem ähnlichen Schema unterliegen können, während instanzbasierte Verfahren Dokumente hauptsächlich inhaltlich auf Gleichheit überprüfen.

Elementebene vs. Strukturebene: Ein Matcher auf Elementebene vergleicht individuelle Schemaelemente und ihre Attribute miteinander, wohingegen ein Matcher auf Strukturebene in Betracht zieht, welche Elemente kombiniert sind, auf welcher Hierarchieebene sich ein Element befindet oder wie viele und welche Kindknoten ein Element besitzt.

linguistisch basiert: Linguistische Methoden schließen einfache String- und auch Teilstringvergleiche ein, beschreiben aber auch komplexere Verarbeitungen wie Tokenisierung und Grundformreduzierung von Wörtern oder das Überprüfen auf Synonym- oder Hyponymität. Bei schemenbasierten Ansätzen bedeutet das das Überprüfen auf Namens- und Beschreibungsähnlichkeiten oder die Berücksichtigung globaler Namespaces. Bei instanzbasierten Verfahren kommen an dieser Stelle Techniken aus dem Information Retrieval zum Einsatz, wie Wortfrequenzen oder Schlüsselterme.

constraint-basiert: Datentypen und Wertebereiche, zum auch Teil Wertemuster, unterscheiden sich oft zwischen Elementen. Davon, sowie von Unterschieden bei der Kardinalität oder Einzigartigkeit von Elementen, profitieren constraint-basierte Ansätze. Auf Strukturebene lässt sich hier auch ein Graph-Matching einsetzen.

externe Information: Manche Verfahren verwenden externe Informationsquellen, wie Wörterbücher bei linguistischen Herangehensweisen. Auch die Wiederverwendung früherer positiver Vergleichsergebnisse ist eine gängige Anwendung (siehe auch [3]).

Matching-Kardinalität: Ähnlich der Outputdimension kann ein Vergleich ein eindeutiges 1:1 Mapping zwischen Elementen ergeben, oder aber ein Element wird mehreren anderen zugewiesen. Letzteres kann sinnvoll sein, um zusammengesetzte Elemente zu erkennen, die auf einer Seite des Vergleichs atomar sind.

individuell vs. kombiniert: Für manche Problemstellungen reicht ein einfacher Matcher, der nur einen Algorithmus einsetzt. Der Trend geht allerdings zu Matchern, die mehrere Verfahren kombinieren. Hier wird unterschieden zwischen ‘hybriden’ und ‘multiplen’ Matchern. Letztere wenden mehrere Verfahren seriell in einer Ausführungskette an, während hybride Ansätze mehrere Verfahren in einem Algorithmus kombinieren.

Dies sind längst nicht alle Einteilungsmöglichkeiten. In [8] wird beispielsweise nach regel- und lernbasierten Ansätzen unterschieden. In dieser Arbeit allerdings wird die oben beschriebene Einteilung vorgenommen und sich darauf beschränkt.

Nachdem diese kurze Übersicht gegeben wurde, werden nun die in Geminus benutzten Verfahren erläutert und in den Aspekten der *Dimensionen* und *Klassifizierung* eingeordnet.

3.2 Verwendete Verfahren

Die Kombination folgender Verfahren ermöglicht eine umfangreiche Berechnung der Ähnlichkeit zweier Paths der verschiedenen Inputs. Jedes Verfahren gibt dabei einen reellen Wert im Intervall $[0, 1]$ aus und lässt sich gewichten. Die Gewichte sind ähnlich der Idee der Kombination von Methoden aus [6]. Der Wertebereich jedes Verfahrens ist dabei erschöpfend zwischen 0 und 1, wobei 0 eine komplette Unähnlichkeit und 1 die Identität auf der entsprechenden Vergleichsebene bedeutet. Es wird jeder Path mit jedem verglichen und ein jeweiliger Ähnlichkeitswert berechnet. Auf den Einfluss der Gewichtung und sinnvolle Werte wird in Abschnitt 4 im Rahmen der Evaluation eingegangen.

Nach [9] ist es sinnvoll, gerade die Mappings der Elementebene nicht durch einen solchen Intervall anzugeben, sondern als Generalisierungen und Teilmengenbeziehungen. Auch in [20] ist eine Hauptaussage, dass Mappings nie direkt passen. Im Kontext dieser Arbeit aber soll ein Ähnlichkeitswert aus der Kombination mehrerer Verfahren berechnet werden, daher werden auch durch die Verfahren, die auf der Elementebene arbeiten, nur Werte zwischen 0 und 1 zugewiesen.

Es handelt es sich hier um einen Algorithmus, welcher keine externen Informationen nutzt, mit Ausnahme der Buchstabenliste zur Normalisierung. Es gibt auch keine Trainingsdaten wie in [7]. Auf die Matching-Kardinalität wird in Abschnitt 3.3 im Detail eingegangen.

Nun werden die einzelnen Verfahren vorgestellt und ihre Arbeitsweisen an einem Beispiel⁵ deutlich gemacht. Ein fiktive XML-Dokumentsammlung A beinhalte die zwei Dokumente

```
<TEI.2><text><body>
  <div1>
    τοῖσι δ' ἀνιστάμενος μετέφη
  </div1>
</body></text></TEI.2>
```

und

```
<TEI.2><text><body>
  <div1>
    πόδας ὠκύς <name>Ἀχιλλεύς</name>·
  </div1>
</body></text></TEI.2>
```

und eine andere Kollektion B beinhalte nur ein Dokument mit folgendem Inhalt.

```
<TEI><text>
  <div>
    Τόν δ' ἀπαμειβόμενος προσέφη πόδας ὠκύς Ἀχιλλεύς·
  </div>
</text></TEI>
```

Betrachtet wird im Folgenden der Vergleich der Paths `TEI.2/text/body/div1` und `TEI/text/div`.

3.2.1 Name Similarity

Als der Name eines Paths wird sein letzter Teilstring betrachtet, also der Tagname in einem XML-Path, beziehungsweise der Spaltenname einer Datenbanktabelle. Der Vergleich der Namen basiert auf der Levensthein-Distanz. Daher ist dieses Verfahren als schemenbasiert, linguistisch und auf Elementebene einzuordnen.

Der Levensthein-Algorithmus berechnet prinzipiell den Unterschied zwischen zwei Zeichenketten und gibt die minimale Anzahl der Veränderungen (Löschen, Einfügen, Ersetzen von Buchstaben) aus, die nötig sind, um die Buchstabenketten aneinander anzugleichen. Wird diese Zahl nun durch die Länge des längeren Vergleichstrings geteilt, ergibt dies ein Maß für die Unähnlichkeit. Die Ähnlichkeit von Wort w_1 und Wort w_2 – die an dieser Stelle die Namen der Paths p_1 und p_2 darstellen – als Zahl im Intervall $[0, 1]$ wird also folgendermaßen berechnet:

⁵Homer, Ilias 1,58 und 1,84, in etwa „Ihm antwortete drauf der mutige Renner Achilleus“, nach Johann Heinrich Voß.

$$sim_{name}(p_1, p_2) = 1 - \frac{levensthein(w_1, w_2)}{max(length(w_1), length(w_2))} \quad (1)$$

Im Beispiel sind die zu vergleichenden Namen `div1` und `div`. Die Levensthein-Distanz beträgt genau 1 (ein Einfügen bzw. Entfernen der Ziffer am Ende, je nach Richtung). Die Ähnlichkeit ist damit $1 - 1/4 = 0,75$.

Die Autoren von [4] und [15] zeigen, dass sich die Namensähnlichkeit und die Editierdistanz noch verfeinern lassen. Hier allerdings ist die Spezialisierung der einzelnen Methoden nicht das Ziel. Die Strings der Namen sind nicht außergewöhnlich komplex und der ursprüngliche Algorithmus ist dafür ausreichend.

3.2.2 Path Depth Similarity

Die Ähnlichkeit der Path- beziehungsweise Pfad-Tiefe ist ein eher simples Verfahren, welches auf die Hierarchie von Elementen Bezug nimmt. Es ist als schemenbasiert und auf der Strukturebene einzuordnen. Die Tiefe oder Länge eines Paths p bezeichne hierbei die Anzahl der slash-separierten Glieder.

$$sim_{path}(p_1, p_2) = \frac{min(length(p_1), length(p_2))}{max(length(p_1), length(p_2))} \quad (2)$$

Es ist unwahrscheinlich, dass sich ähnliche Paths in der Tiefe stark unterscheiden, wie zum Beispiel ein Wurzelknoten mit einem tiefliegenden Tag, welcher auf eine Lücke im Text hinweist.

Im Beispiel hat der Path aus Dokument A die Tiefe 4, der aus Dokument B nur 3. Die Ähnlichkeit beträgt damit $3/4 = 0,75$.

3.2.3 Cosine Similarity

Das Vektorraummodell ist ein wesentlicher Teil des Algorithmus. Es wird u.a. auch in [2] und [12] beschrieben. Zu den Paths p_1 und p_2 werden Wortvektoren v_{p1} und v_{p2} gebildet, welche mit dem Kosinusmaß verglichen werden. Dazu wird das Skalarprodukt der Vektoren durch das Produkt ihrer euklidischen Norm (Länge) geteilt. Der Vektorraum ist n -dimensional, wobei n die Anzahl der insgesamt auftretenden Wörter ist.

$$sim_{cos}(p_1, p_2) = \frac{v_{p1} \cdot v_{p2}}{|v_{p1}| \cdot |v_{p2}|} \quad (3)$$

Da der Winkel zwischen den Vektoren 90° nicht überschreiten kann (weil die Termfrequenzen nicht negativ sein können), liegt das Ergebnis im Intervall $[0, 1]$.

Oft werden die Frequenzen der Terme für die Vektoren benutzt, hier allerdings wird für die Inhalte der Vektoren das TF-IDF-Gewicht berechnet (auch diese werden nicht negativ, siehe [17, 2, 12]). Dazu werden die Paths als Dokumente betrachtet und die Termfrequenz wird auf die Anzahl aller Wörter im jeweiligen Path (Tokens) normiert.

Für Wort i , Path j , $n_{i,j}$ die Frequenz von Wort i in Path j und P die Menge der Paths p ist also

$$(tf\ idf)_{i,j} = tf_{i,j} \cdot idf_i = \frac{n_{i,j}}{\sum_k n_{k,j}} \cdot \log \frac{|P|}{|\{p | w_i \in p\}|} \quad (4)$$

Ein hohes TF-IDF-Gewicht wird durch eine hohe Termfrequenz in einem Path erreicht, während niedrige Dokumentfrequenzen hohe Werte begünstigen. Durch die Gewichtung sollen folglich Terme durch niedrige Werte „gefiltert“ werden, die selten sind oder in übermäßig vielen Paths auftreten.

Die Kosinusähnlichkeit ist damit instanzbasiert und auf der Elementebene. Dadurch, dass die Wörter durch eventuelle Betacode-Konvertierung, Tokenisierung und Normalisierung vorverarbeitet werden, kann man auch von einem linguistisch basierten Ansatz sprechen.

Für unser Beispiel ergibt sich also nach Tokenisierung und Normalisierung folgende Wortliste:

$W = \{0 \rightarrow \tau\omicron\iota\sigma\iota, 1 \rightarrow \tau\omicron\nu, 2 \rightarrow \delta, 3 \rightarrow \alpha\nu\iota\sigma\tau\alpha\mu\epsilon\nu\omicron\varsigma, 4 \rightarrow \alpha\pi\alpha\mu\epsilon\iota\beta\omicron\mu\epsilon\nu\omicron\varsigma, 5 \rightarrow \mu\epsilon\tau\epsilon\phi\eta, 6 \rightarrow \pi\rho\omicron\sigma\epsilon\phi\eta, 7 \rightarrow \pi\omicron\delta\alpha\varsigma, 8 \rightarrow \omega\kappa\upsilon\varsigma, 9 \rightarrow \alpha\chi\iota\lambda\lambda\epsilon\upsilon\varsigma\}$

Die reinen Frequenzvektoren v_{A_f} und v_{B_f} wären damit:

$v_{A_f} = [1, 0, 1, 1, 0, 1, 0, 1, 1, 1]$ und

$v_{B_f} = [0, 1, 1, 0, 1, 0, 1, 1, 1, 1]$

und es würde sich allein damit durch das Kosinusmaß eine Ähnlichkeit von 0,5714286 ergeben. Eine TF-IDF-Gewichtung würde zwar an dieser Stelle noch durchgeführt werden, ist allerdings in diesem Minimalbeispiel nicht sinnvoll, da die Textmenge zu gering ist.⁶ An dieser Stelle soll es zur Verdeutlichung genügen.

3.2.4 Dice Similarity

Dieses Ähnlichkeitsmaß basiert auf dem Dice-Koeffizienten. Es berechnet das Verhältnis der Wörter (Types), die in beiden Paths vorkommen zu allen vorkommenden und ist damit ebenfalls instanzbasiert und auf Elementebene.

Sei W_{p_1} die Menge der Wörter aus einem Path p_1 eines Inputs und W_{p_2} die eines anderen Paths p_2 , dann lässt sich die Ähnlichkeit wie folgt berechnen:

$$sim_{dice}(p_1, p_2) = \frac{2|W_{p_1} \cap W_{p_2}|}{|W_{p_1}| + |W_{p_2}|} \quad (5)$$

⁶Die Ähnlichkeit würde nur noch rund 0,025 betragen.

Für die Paths aus dem Beispiel ergibt sich so eine Ähnlichkeit von $8/14 = 0,571428571$. Dadurch, dass alle Frequenzen 1 sind, erinnert das Ergebnis stark an das Ergebnis des Kosinus-Vektorvergleiches.

3.2.5 Absolute Frequency Similarity

Mit der Annahme, dass äquivalente Paths auch gleich häufig auftreten, arbeitet dieser schemen- und constraint-basierte Ansatz auf Elementebene. Die Häufigkeiten der Paths werden gezählt und ihr Verhältnis ergibt das Maß für die Ähnlichkeit. Bei Datenbanken wird die Anzahl der Zeilen gezählt.

Seien nun f_{p1} die Frequenz für p_1 und f_{p2} die Frequenz für p_2 . Dann ist

$$sim_{freq\ abs}(p_1, p_2) = \frac{\min(f_{p1}, f_{p2})}{\max(f_{p1}, f_{p2})} \quad (6)$$

Im Beispiel tritt `TEI.2/text/body/div1` zweimal auf, `TEI/text/div` nur einmal. Dadurch ergibt sich hier ein Wert von 0,5.

3.2.6 Relative Frequency Similarity

Es ist denkbar, dass die Frequenzen der Paths auf die Dokumentzahl normiert werden müssen, etwa wenn die Inputs unterschiedliche Ausmaße haben oder auf unterschiedliche Mengen von Dateien aufgeteilt sind. Bei Datenbanken wird von einer „großen“ Datei ausgegangen.

Sei nun d_i die Anzahl der Dateien, aus denen der Input besteht, der p_i enthält, $i \in \{1, 2\}$. Die Ähnlichkeit der relativen Häufigkeit ist dann

$$sim_{freq\ rel}(p_1, p_2) = \frac{\min(\frac{f_{p1}}{d_1}, \frac{f_{p2}}{d_2})}{\max(\frac{f_{p1}}{d_1}, \frac{f_{p2}}{d_2})} \quad (7)$$

Im Beispiel wird so nun 1 erreicht, da die zu vergleichenden Paths im Verhältnis zu ihren Dokumenten gleich häufig auftreten. Es ist nicht angedacht, die absolute und die relative Häufigkeit gleichzeitig zu benutzen. Je nach Input sollte höchstens eine mittels Gewichtung in das Ergebnis des Vergleiches eingehen.

3.2.7 Content Type Similarity

Um Ähnlichkeiten zwischen Paths, die hauptsächlich Zahlen enthalten, mit Paths, die annehmend nur Wörter beinhalten, auszuschließen, eignet sich dieses recht simple constraint- und instanzbasierte Verfahren auf Elementebene.

Es stellt für jeden Path ein Verhältnis von Zahlen zu Nichtzahlen auf und vergleicht diese. Dazu wird für jedes Wort geprüft, ob es sich um eine ganze Zahl handelt, insbesondere

auch, wenn Zahlen für die Wortliste ignoriert werden. Relevanz ergibt sich bei Paths, die möglicherweise IDs, oder aber auch Geldbeträge enthalten, sowie Auflistungen von gezählten Waren oder Personen.

Sei r_{p1} das Verhältnis von Zahlen zu allen auftretenden Wörtern in p_1 und r_{p2} analog.

$$\text{sim}_{\text{content}}(p_1, p_2) = 1 - |r_{p1} - r_{p2}| \quad (8)$$

Obiges Beispiel enthält zu 100% Wörter, so dass der Vergleich auch hier 1 ergibt.

Es wird erkennbar, dass viele der Methoden simpel gehalten wurden. In [10] und [19] werden Verfahren vorgeschlagen, die mit komplexen Schlüsselbeziehungen und Konzepten Datenbanken untereinander sehr genau aufeinander abbilden können, oder auch Verfahren, die durch die Auswertung einer erweiterten Struktur Ähnlichkeiten in XML-Dokumenten finden. Ziel dieser Arbeit aber ist es, auch zwischen Datenbanken und XML-Dateien Struktur mappings zu finden und daher Verfahren zu verwenden, die nach Möglichkeit auf eine Vielzahl von verschiedenen Inputs anwendbar sind.

3.3 Ausgabe

Nachdem jeder Path aus dem ersten Input mit jedem aus dem zweiten Input verglichen wurde, gibt Geminus seine Vergleichsergebnisse in zwei Dateien aus. Dazu kommt eine GUI-Tabelle zur visuellen Übersicht.

In einer detaillierten Datei stehen sämtliche Vergleichsergebnisse, die mindestens einen beliebigen, aber festen Wert erreicht haben. Sie beinhaltet zeilenweise jeweils tabsepariert die beiden verglichenen Paths, deren berechnete Gesamtähnlichkeit und dazu die Ergebnisse der einzelnen angewandten Verfahren mit ihren jeweiligen Gewichten, mit denen sie in das Gesamtergebnis eingingen. Eine solche Ausgabe ist als Anlage D auf Seite 45 zu finden.

In einer weiteren, weniger ausführlichen Datei werden alle *eineindeutigen* Vergleichsergebnisse gespeichert. Zwei Paths p_A und p_B sind hierbei eineindeutig ähnlich, wenn p_A aus Input 1 sein bestes Vergleichsergebnis bei p_B aus Input 2 erreicht hat und der beste Matching-Partner für p_B auch p_A ist.

Die Übersicht in einer grafischen Tabelle am Ende des Algorithmus ist optional. Sie stellt für den menschlichen User allerdings die einfachste Möglichkeit dar, die Ergebnisse in einem Blick zu erfassen. In einer Matrix werden alle Paths aus Input 1 denen aus Input 2 gegenüber gestellt und

	dating	geography
head	0,44095	0,48935
date	0,77802	0,17866
placeName	0,17860	0,70370

Abbildung 3: Ein Ausschnitt aus einer JTable. Das Element „head“ ist deutlich als Container zu erkennen.

die Zellen mit den Vergleichsergebnissen werden je nach Sicherheit für eine Ähnlichkeit mehr oder weniger kräftig grün eingefärbt. Per Mouseover auf einzelne Zellen erhält man

dann weitere Details über die Ergebnisse der einzelnen Vergleichsmethoden. Eine solche vollständige Tabelle ist als Anlage (C) zu finden.

Wie in Abbildung 3 zu erkennen ist, können rein optisch aus der Tabelle auch Informationen entnommen werden, die in den ‘nackten’ Zahlen möglicherweise nur schwer zu entdecken sind. Sie zeigt einen Ausschnitt des Vergleichsergebnisses von den Epiduke-XML-Dateien mit der Epiduke-Datenbank. Die XML-Elemente „date“ und „placeName“ passen offenbar gut zu „dating“ und „geography“ in der Datenbank. Die noch recht hohe Ähnlichkeit des Elementes „head“ zur Datenbank an dieser Stelle lässt sich so erklären, dass es in der XML-Hierarchie die Elemente „date“ und „placeName“ enthält und somit auch eine gewisse und natürlich geringere Ähnlichkeit zu den Datenbank-Paths besitzt.

Für die Output-Dimension und die Matching-Kardinalität bedeutet das also, dass je nach Betrachtungsweise sowohl ein Sicherheitsmaß für einzelne Path-Paare die Ähnlichkeit beschreibt, als auch eineindeutige Beziehungen binär angegeben werden. Das heißt, die Zuordnungen finden sowohl n:1 als auch 1:1 statt. Einem Path können mehrere andere Paths zugewiesen werden, auch mit ähnlich hohen Vergleichsergebnissen. Aber nur ein Vergleichsergebnis ist das höchste und eventuell liegt eine eineindeutige Beziehung zu dem Matching-Partner vor.

Zusätzlich lassen sich im Ansatz auch durch Betrachten der Tabelle Teilmengenbeziehungen erkennen – also die Zuordnung von Elementen zu zusammengesetzten Elementen – allerdings ist das ein händischer Prozess und nichts, was das Programm tatsächlich als Ergebnis vorschlägt.

Wie gut nun die Methoden für sich alleine und zusammen arbeiten, wird im folgenden Abschnitt umfangreich analysiert. Zunächst werden die Daten aus Abschnitt 2 mit gleichverteilten Gewichten verglichen, um dann herauszufinden, welche Verfahren sich für welche Input-Formate eignen und welche nicht. Anschließend werden die erfolgreichen Verfahren entsprechend gewichtet, damit ein erneuter Vergleich die erzielten Verbesserungen der Ergebnisse zeigen kann.

4 Evaluation

Im Allgemeinen ist es problematisch zu sagen, wie gut ein Algorithmus oder Programm funktioniert, insbesondere im Vergleich mit anderen. Wie in Abschnitt 3.1 gezeigt, gibt es sehr viele Punkte, in denen sich die Verfahren unterscheiden können. In erster Linie kommt es bei allen Schema-Matching-Verfahren darauf an, wie in [5] beschrieben, Zeit und Mühen durch Automatisierung zu sparen. Die Verwendung des Programms sollte bestenfalls möglich sein, ohne sonderlich viel zusätzliche Zeit darin investieren zu müssen, eine geeignete Konfiguration zu finden und zu erstellen, oder nachträglich die Ergebnisse zu säubern.

Vorbereitender Aufwand: Dazu gehört zum Beispiel das Training eines machine-learning basierten Ansatzes oder die Vor- und Aufbereitung von externen Informationsquellen, die eventuell benötigt werden. Für Geminus müssen an dieser Stelle nur verschiedene Parameter und Gewichte gesetzt werden. Auf die Auswirkungen dieser wird später eingegangen.

Aufwandseinsparung: Diese ist schwer in Zahlen festzulegen. Oft kommen nachträgliche Arbeiten hinzu, wie das Korrigieren und Verbessern der erzeugten Mappings, also Entfernen von überflüssigen oder das Suchen von fehlenden Ergebnissen. Abhängig vom Input lässt sich oft erst im Nachhinein einschätzen, wie viel Arbeit durch den Einsatz des Programms effektiv gespart oder wie viel Sicherheit hinzu gewonnen wurde. Für mehrere tausend XML-Dateien zu überprüfen, ob die gleichen Paths regelmäßig auftreten und immer den gleichen Typ von Information beinhalten, um diese dann mit mehreren Datenbanktabellen abzugleichen, ist sicherlich keine angenehme manuelle Arbeit. Für eine entsprechende Automatisierung wird die Zeit für die Vor- und Nachbereitung hier in Kauf genommen.

Die Qualität der Ergebnisse ist selbstverständlich ebenfalls ein wesentlicher Anspruch an ein solches Programm. Dieser Abschnitt wird ausführlich zeigen, wie gut Geminus tatsächlich auf Textdaten arbeitet. Dazu wurden die Daten aus Abschnitt 2 in folgenden Kategorien miteinander verglichen:

- Datenbank versus Datenbank
 - Perseus-DB versus Epiduke-DB
- Datenbank versus XML
 - Epiduke-DB versus Epiduke-XML
 - PHI7-DB versus Epiduke-XML
 - MISC-DB versus Epiduke-XML
- XML versus XML
 - Perseus-XML versus Epiduke-XML
 - American History versus Richmond Times

In den Datenbanken werden wie in Abschnitt 2 beschrieben jeweils die Tabellen „authors“, „works“ und „sentences“ gewählt.

Um die Ergebnisse auswerten und beurteilen zu können, muss zunächst ein gewisser Erwartungswert bestehen. Für jeden Vergleich wird also im Voraus von Hand ein *Goldstandard* festgelegt, der die gewünschten Zielmappings enthält, die das Programm finden soll.

Ein gewünschtes Mapping zählt als gefunden, wenn die Mapping-Partner beidseitig unter den besten zwei Ergebnissen für den jeweiligen Path liegen. Allerdings ist im Goldstandard hier für jeden Path nur ein Partner vorgesehen. Hat also beispielsweise ein Path in einem 1:n-Mapping mehrere äquivalente Partner, gelten n-1 Mappings als nicht erwünscht.

Gefundene Mappings, die im Goldstandard festgelegt wurden, werden wie in [5] als *true positives* bezeichnet. Alle weiteren gefundenen Mappings, die außerhalb des Goldstandards liegen, werden folglich *false positives* genannt. Fehlen Mappings, die im Goldstandard stehen, so sind diese *false negatives*. Korrekterweise nicht gefundene Mappings heißen hingegen *true negatives*.

Es wird nun folgendermaßen vorgegangen und gezählt: Zunächst wird untersucht, welche der Path-Paare aus der Musterlösung auch gefunden wurden. Der maximal erreichte Vergleichswert wird notiert. Nun gilt alles, was bis zu 50% von diesem besten Wert erreicht hat, ebenfalls als gefunden. Alle Mappings mit entsprechend hohen Vergleichsergebnissen, die nicht im Goldstandard vorkommen, sind nun false positives. Erwünschte Ergebnisse, welche die 50% des Besten nicht erreichen, sind false negatives.

Eine Grenze ist notwendig, da im Output jedes Mapping mit einem Wert erscheint, ist dieser auch noch so gering. Die 50% sind dabei willkürlich gesetzt. Sie sollen verhindern, dass sehr niedrige Werte noch als positives gelten, wobei eine Relativität zum besten Ergebnis gewahrt wird. Ein einfacher Schwellenwert wie „Alles unter 0,4 ist nicht gefunden“ würde zum Beispiel keinen Unterschied machen zwischen einem Bestwert von 0,9 und einem von 0,6, und damit unter anderem den Einfluss der Qualität der Daten vernachlässigen.

Betrachtet und analysiert werden beide textuellen Outputs, also die vollständigen n:1-Mappings und die eineindeutigen 1:1-Mappings, wobei nur letztere tabellarisch aufgeführt werden.

Um die Funktionalität der einzelnen eingesetzten Verfahren zu beobachten, werden diese zunächst alle mit dem gleichen Gewicht⁷ eingesetzt. Es wird überprüft, was jeweils funktioniert, und was nicht. Anschließend wird versucht, nützliche Gewichte zu finden, und ausgewertet, wie stark die Ergebnisse sich dadurch verbessern.

Folgende Settings werden verwendet:

⁷Bei sieben Verfahren geht jedes mit rund 14,28% ins Ergebnis ein.

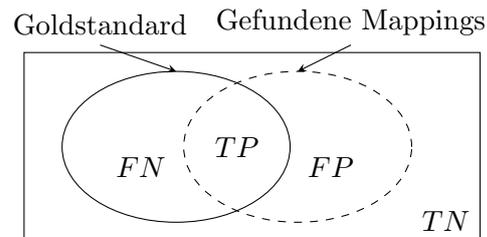


Abbildung 4: Das Ziel ist eine möglichst große Überlagerung der gefundenen Mappings mit dem Goldstandard.

- Keine Zahlen werden ignoriert
- Keine Stoppwörter werden entfernt
- Keine seltenen Wörter werden entfernt
- Betacode-Konversion nur, wenn nötig
- Griechische Buchstaben werden normalisiert
- XML-Pfade müssen in mindestens 50% der Dateien präsent sein

Als Abschluss der Evaluation wird dann untersucht, ob und wie stark sich das Entfernen von seltenen und Stoppwörtern, das Ignorieren von Nummern, die Ersatzkodierung ‘BetaCode’ und die Normalisierung auf die instanzbasierten Verfahren auswirken.

4.1 Ungewichtet

DB-DB – Perseus versus Epiduke

Da die Datenbanken im selben Schema vorliegen, ist das zu findende Mapping eher trivial. Als Goldstandard werden Mappings wie in Tabelle 1 festgelegt.

Perseus/authors/dating	Epiduke/authors/dating
Perseus/authors/female	Epiduke/authors/female
Perseus/works/corpora_author_id	Epiduke/works/corpora_author_id
Perseus/works/work_id	Epiduke/works/work_id
Perseus/sentences/s_id	Epiduke/sentences/s_id
Perseus/sentences/sentence	Epiduke/sentences/sentence

Tabelle 1: Goldstandard von Perseus-DB versus Epiduke-DB.

Nach der ungewichteten Berechnung der Ergebnisse treten die eineindeutigen Mappings auf, wie in Tabelle 2 zu sehen.

Perseus	Epiduke	Ähnlichkeit
authors/dating	authors/dating	0.66621417
authors/female	authors/female	0.7622915
works/corpora_author_id	works/corpora_author_id	0.90269667
works/work_id	works/work_id	0.7331666
sentences/s_id	sentences/s_id	0.6448523
sentences/sentence	sentences/sentence	0.7148659

Tabelle 2: Vergleichsergebnis von Perseus-DB versus Epiduke-DB.

Von den insgesamt 169 Mappings sind nun per Definition alle mit einem Wert über 0.451348335 false positives, abzüglich der 6 oben genannten true positives. Es gibt an dieser Stelle 43 false positives, allerdings ist kein Mapping von diesen eineindeutig.

DB-XML – Epiduke versus Epiduke

In den Datenbanken lassen sich die Informationen wie in Tabelle 3 gezeigt in den XML-Dateien wiederfinden und gelten somit als Goldstandard.

Epiduke/authors/author	TEI/teiHeader/fileDesc/titleStmt
Epiduke/authors/dating	TEI/text/body/head/date
Epiduke/authors/geography	TEI/text/body/head/placeName
Epiduke/sentences/sentence	TEI/text/body

Tabelle 3: Goldstandard von Epiduke-DB versus Epiduke-XML.

Das Programm gibt die in Tabelle 4 aufgeführten 1:1 Mappings aus, wobei schräger Text auf ein false positive hinweist.

Epiduke-DB	Epiduke-XML	Ähnlichkeit
authors/author	TEI/teiHeader/fileDesc/titleStmt	0.69374335
authors/dating	TEI/text/body/head/date	0.72737134
<i>authors/epithets</i>	<i>TEI/teiHeader/revisionDesc</i>	<i>0.4637525</i>
authors/geography	TEI/text/body/head/placeName	0.65607005
sentences/sentence	TEI/text/body	0.47416136

Tabelle 4: Vergleichsergebnis von Epiduke-DB versus Epiduke-XML.

Insgesamt gibt es 338 Mappings. Vier von den fünf eindeutigen Mappings sind true positives, eines ist ein false positive⁸. Auch alle Ergebnisse über einem Wert von mindestens 0.36368567 gelten als false positives. Abzüglich der 4 true positives sind das 130, und nur eines davon ist eindeutig.

DB-XML – PHI7 versus Epiduke

Das PHI7-Korpus enthält neben der DDbDP in Griechisch auch lateinische und sogar etwas hebräische und koptische Texte, ein schlechteres Vergleichsergebnis ist also zu erwarten. Neben vermuteten textuellen Übereinstimmungen ist inhaltlich nichts weiter in der Datenbank zu finden, daher ist der Goldstandard nur das in Tabelle 5 angegebene Mapping.

PHI7/sentences/sentence	TEI/text/body
-------------------------	---------------

Tabelle 5: Goldstandard von PHI7-DB versus Epiduke-XML.

Die berechneten eindeutigen Mappings sind nun wie in Tabelle 6 zu sehen⁹.

⁸Anmerkung: Die Datenbankspalte ‘epithets’ enthält nur leere Strings.

⁹„publicationStmt“ sei hier durch „publicStmt“ abgekürzt.

PHI7-DB	Epiduke-XML	Ähnlichkeit
<i>authors/author</i>	<i>TEI/teiHeader/fileDesc/publicStmnt/authority</i>	0.3267445
<i>authors/epithets</i>	<i>TEI/teiHeader/revisionDesc</i>	0.3219797
<i>sentences/sentence</i>	<i>TEI/text/body</i>	0.3648154

Tabelle 6: Vergleichsergebnis von PHI7-DB versus Epiduke-XML.

Es sind also zwei false positives dabei. Eines erinnert an den Vergleich mit der Epiduke-Datenbank zuvor und das andere unterliegt offenbar der hier irreführenden hohen Namensähnlichkeit. Neben diesen zwei eindeutigen false positives gibt es noch eine Reihe weiterer, und zwar alle mit einem Wert über 0.1824077. Da ein solch geringer Wert recht leicht zu erreichen ist, sind hier von den 338 Mappings ganze 204 false positives.

DB-XML – MISC versus Epiduke

Das MISC-Korpus ist ein griechisches Korpus. Es gibt allerdings inhaltlich theoretisch keine Übereinstimmungen, bis auf die gleiche Sprache. Ein Goldstandard mit dem Mapping von „MISC/sentences/sentence“ zu „TEI/text/body“ wäre hier also nur rein strukturell motiviert.

MISC	Epiduke-XML	Ähnlichkeit
<i>authors/author</i>	<i>TEI/teiHeader/fileDesc/publicStmnt/authority</i>	0.3264982
<i>sentences/sentence</i>	<i>TEI/teiHeader/revisionDesc/change</i>	0.3952415

Tabelle 7: Vergleichsergebnis von MISC-DB versus Epiduke-XML.

Obwohl die Ähnlichkeiten eher gering sind, werden trotzdem falsche, eindeutige Mappings gefunden, wie in Tabelle 7 aufgelistet.

XML-XML – Perseus versus Epiduke

Bei dem Vergleich von XML-Dateien mit ihrer Vielzahl verschiedener Tags ist es schwierig, sich auf einen Goldstandard festzulegen. Insbesondere muss abgewogen werden, ob man gleiche Strukturelemente finden will, in denen aber nicht unbedingt auch der gleiche Inhalt stehen muss, oder ob man Elemente mit gleichem Inhalt aufeinander mappen möchte, auch, wenn sie in unterschiedlichen Strukturelementen zu finden sind. Soll also beispielsweise ein Titel-Path gefunden werden, der in beiden Inputs steht, auch wenn im Voraus klar ist, dass er jeweils unterschiedliche Strings enthält? Diese Frage wird in Abschnitt 4.2 ausführlich diskutiert. Hier sei zunächst der Goldstandard wie in Tabelle 8 gegeben.

Es fällt auf, dass es inhaltlich äquivalente Paths gibt, in denen zu tieferen bzw. höheren Ebenen kaum ein qualitativer Unterschied besteht; zum Beispiel ist der Inhalt von „lang-Usage“ der gleiche, wie von „langUsage/language“, ähnlich ist es bei „TEI/text“ und „TEI/text/body“.

TEI.2	TEI
teiHeader	teiHeader
teiHeader/fileDesc	teiHeader/fileDesc
teiHeader/fileDesc/titleStmt	teiHeader/fileDesc/titleStmt
teiHeader/fileDesc/titleStmt/title	teiHeader/fileDesc/titleStmt/title
teiHeader/fileDesc/publicationStmt	teiHeader/fileDesc/publicationStmt
teiHeader/fileDesc/sourceDesc	teiHeader/fileDesc/sourceDesc
teiHeader/profileDesc	teiHeader/profileDesc
teiHeader/profileDesc/langUsage	teiHeader/profileDesc/langUsage
teiHeader/profileDesc/langUsage/language	teiHeader/profileDesc/langUsage/language
text	text
text/body	text/body
text/body/div1/head	text/body/head
text/body/div1/head/date	text/body/head/date
text/body/div1/head/placeName	text/body/head/placeName

Tabelle 8: Goldstandard von *Perseus-XML* versus *Epiduke-XML*.

Die ohne Gewichte berechneten eindeutigen Mappings sind als Anlage A zu finden.

Unter den 1040 Mappings sind also 14 true positives, und ein Mapping wurde nicht eindeutig gefunden und ist daher false negative. 279 gelten mit einem Wert über 0.387962135 als false positives, wovon 3 eindeutig sind.

XML-XML – American History versus Richmond Times

Zur Abgrenzung werden englischsprachige Korpora miteinander verglichen. Da diese sich inhaltlich nicht allzu sehr überschneiden sollten, werden gute Ergebnisse hauptsächlich von den strukturellen Verfahren erwartet. Semantisch gleiche Paths sollen also aufeinander abgebildet werden, insbesondere der eigentliche Textcontainer.

Bei einer Eingabegröße von jeweils rund 100MiB entstehen 69 eindeutige Mappings, von 0.79905044 bis 0.6718455. Dabei werden die XML-Strukturen offenbar weitestgehend gut aufeinander abgebildet, wie zum Beispiel „TEI.2/text/body“ und „TEI.2/text/body“ mit 0.7988492.

Interessant sind Mappings wie „TEI.2/text/body/div1/p/quote/p“ und „TEI.2/text/body/div1/div2/div3/cit/q/p“, mit einem Ähnlichkeitswert von 0.7600756, welche offenbar eine Beziehung zwischen Paragraphen in quotes und citations zeigen.

Insgesamt sind die Ergebnisse durchaus verbesserungswürdig. Nachfolgend wird untersucht, wie sich die Gewichte sinnvoll setzen lassen und welchen Einfluss sie tatsächlich auf die Ergebnisse nehmen können.

4.2 Wahl der Gewichte

Zunächst fällt auf, dass die verschiedenen Maße in Abhängigkeit vom Input verschiedene Wirkungen zeigen. Daher werden sie für die drei Input-Kategorien getrennt untersucht.

DB-DB

name similarity: Durch die identischen Datenbankschemata sind die Werte bei den true positives auf natürliche Weise hoch. Allerdings bekommen auch ein paar wenige false positives ganz gute Werte, weil sie wie bei „authors/corpora_author_id“ und „works/corpora_author_id“ gleich heißen, oder wie bei „s_id“ und „a_id“ noch recht ähnlich sind, obwohl sie nicht das gleiche beinhalten. Dies sei beispielhaft durch Abbildung 5 grafisch dargestellt.

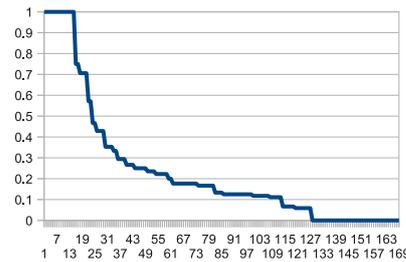


Abbildung 5: Die Namensähnlichkeiten pro Mapping beim Vergleich von Perseus-DB mit Epiduke-DB.

path depth similarity: Dieses Verfahren stellt sich als unsinnig bei der Verwendung mit Datenbanken heraus, da per Definition alle Paths die Tiefe ‘drei’ haben.

cosine similarity: Der Vektorvergleich erreicht an dieser Stelle die höchsten Werte bei den false positives. Der Grund dafür ist, dass viele Tabellenspalten einen leeren String oder „NULL“ enthalten. Die true positives erreichen auch relativ hohe Werte, allerdings ebenso ein paar weitere false positives, da die Autornummern und die Werk- und Satznummern als ähnlich angesehen werden.

dice similarity: Dieses Verfahren verhält sich sehr ähnlich zum Vektorvergleich mit dem Kosinusmaß.

absolute frequency similarity: Die Ergebnisse scheinen zufällig verteilt und lassen keine Auskunft über true und false positives zu.

relative frequency similarity: Da bei Datenbanken jeweils von nur einem Dokument ausgegangen wird, erscheinen die Ergebnisse hier genauso zufällig wie bei der absoluten Frequenz.

content type similarity: Alle Path-Paare, die IDs enthalten, bekommen hohe Werte. Die true positives haben ebenfalls hohe Werte, genauso aber auch einige false positives.

In diesem Fall ist es also ratsam, nur die Namens-, Kosinus-, Dice- und die Content-Type-Ähnlichkeit einzusetzen. Die Gewichte werden wie folgt gesetzt:

- 50% name similarity
- 20% cosine similarity
- 15% dice similarity
- 15% content type similarity

so dass zur Hälfte die Struktur und zur anderen die Inhalte der Instanzen betrachtet werden.

DB-XML

name similarity: Bis auf „date“ und „dating“ gibt es keine sinnvolle Übereinstimmungen. Es gibt sogar Irreführungen bei „authors“ und „authority“.

path depth similarity: Hier ist keine Aussage möglich, da die Datenbank-Paths alle die Tiefe ‘drei’ haben. Es wird bei den äquivalenten Textknoten sogar „TEI/body/text“ gegenüber tieferen unbegründet bevorzugt.

cosine similarity: Es gibt hohe Werte zwischen „sentences“ und potentiell äquivalenten Textpaths, und auch bei „date“ und „dating“. False positives sind in ihrer Zahl gering. Zum Teil verursachen Meta-Annotationen allerdings mitunter ungünstige Ergebnisse, da der Dateiname oft auch Titel, Autor und Autor-ID ist.

dice similarity: Funktioniert gut im Header und auch bei dem eigentlichen Textmapping. Es gibt einen direkten Abfall im Ähnlichkeitswert bei true negatives.

absolute frequency similarity:

Die Werte haben meist keine Aussage, da sie nur langsam abnehmen und dadurch viele false positives dabei sind. Nur bei dem Epiduke-Vergleich gibt es im Header (wie „date“, „placeName“) gute Übereinstimmungen, da für jede XML-Datei eine eigene Datenbankzeile existiert.

relative frequency similarity: Ist hier unbrauchbar, da die Datenbank nur ein Dokument hat. So wird der höchste auftretende Wert circa 0,03.

content type similarity: Über die Hälfte der Mappings erreicht 0,9, danach gibt es einen starken Abfall. Dieses Maß schließt nur eine Ähnlichkeit von Zahlen, wie IDs, auf der Datenbankseite mit textuellen XML-Elementen aus, beinhaltet aber noch viele false positives.

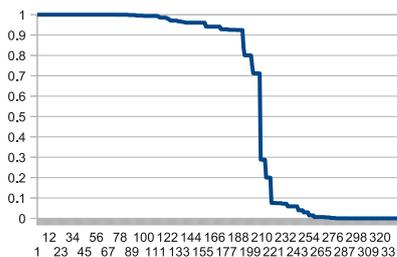


Abbildung 6: Die content type similarity pro Mapping beim Vergleich von Epiduke-DB mit Epiduke-XML.

Bei dem Vergleich von Datenbanken mit XML-Dokumenten erweisen sich viele Maße als nutzlos, allerdings liefern die instanzbasierten Verfahren, die mit den auftretenden Wörtern arbeiten, sehr nützliche Ergebnisse. Daher sollten die Kosinus- und die Dice-Ähnlichkeit beide mit 45% eingehen und die verbleibenden 10% unter Namens- und Content-Type-Ähnlichkeit aufgeteilt werden:

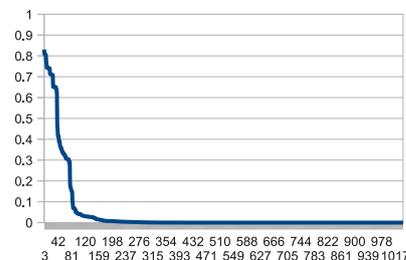
- 45% cosine similarity
- 45% dice similarity
- 5% name similarity
- 5% content type similarity

XML-XML

name similarity: Durch die TEI-Norm haben viele gleiche Paths auch den gleichen Namen. Nennenswerte Ausnahmen sind hier „p“ mit „ab“, „div“ mit „div1“ und „TEI“ mit „TEI.2“. False positives sind allerdings auch darunter, wie Tags mit gleichen Namen trotz stark unterschiedlicher Pfadtiefe oder „fileDesc“ und „profileDesc“.

path depth similarity: Sehr viele Paths sind auf ähnlichen Leveln, daher gibt es hier viele hohe Werte auch bei false positives. Nur stark unterschiedliche Paths, wie zum Beispiel der Wurzelknoten mit einem stark verschachteltem Element, lassen sich ausschließen.

cosine similarity: Die höchsten Werte bekommen Mappings von äquivalenten Paths, danach kommen false positives, zum Beispiel von Verschachtelungen im Header wie in Abbildung 3 auf Seite 18. Darauf folgen mit einem starken Wertabfall die true negatives, wie in Abbildung 7 angedeutet.



dice similarity: Auch hier treten false positives mit relativ geringen Werten auf, wohingegen true positives die höchsten Werte haben.

Abbildung 7: Die Vektorähnlichkeit pro Mapping beim Vergleich von Perseus-XML mit Epiduke-XML.

absolute frequency similarity: Diese ist hier unbrauchbar. Hohe Werte bekommen uninteressante, verschachtelte Paths, während das erste true positive sehr spät auftritt und einen schlechten Wert hat.

relative frequency similarity: Ein Drittel der Mappings hat hohe Werte, dann gibt es einen schnellen Abfall der Werte. Bis auf „placeName“ sind alle Mappings des Goldstandards mit hohen Werten vertreten.

content type similarity: Prinzipiell ist dieses Maß wenig brauchbar, da die meisten

Mappings noch einen Wert von über 0,7 haben. Die true positives sind allerdings darunter auch alle mit hohen Werten vertreten.

Die verschiedenen Verfahren sollten bei dem Vergleich von XML-Dokumenten zusammenspielen. True negatives haben vermutlich nicht ähnliche Namen und gleichzeitig einen ähnlichen Path und eine ähnliche Häufigkeit. Dadurch hat jedes Maß zu einem gewissen Grad seine Berechtigung, abgesehen von der absoluten Frequenz. Im nächsten Schritt werden die Gewichte daher folgendermaßen gesetzt:

- 25% cosine similarity
- 25% dice similarity
- 20% name similarity
- 10% path depth similarity
- 10% relative frequency similarity
- 10% content type similarity

Alle Gewichte wurden intuitiv gewählt. Verfahren, die eine klare Grenze zwischen true positives und true negatives ziehen konnten, bekamen die höchsten Gewichte. Verfahren, die für sich alleine stehend nur bedingt aussagekräftig waren, wurden mit geringeren Gewichten versehen. Modifikationen an diesen Gewichten vermögen sicherlich den Ähnlichkeitswert für das eine oder andere Path-Paar zu erhöhen, möglicherweise dann aber auf die Kosten von anderen Mappings. Das Finden eines absoluten Optimums in Hinsicht auf den Goldstandard ist nicht Ziel dieser Arbeit.

Allgemein hat sich herausgestellt, dass die absolute Frequenz-Ähnlichkeit unbrauchbar ist und auch für die relative eignet sich ein Einsatz nur bei den XML-XML-Vergleichen. Neben diesem Ähnlichkeitsmaß ist auch die Pfadtiefen-Ähnlichkeit nur bei XML-XML-Vergleichen sinnvoll einsetzbar.

4.3 Gewichtet

Nachfolgend wird gezeigt, wie stark sich die Ergebnisse durch eine an das Problem angepasste Gewichtung der Verfahren verbessert haben, jeweils orientiert am selben Goldstandard.

DB-DB – Perseus versus Epiduke

Nach der gewichteten Berechnung der Ergebnisse treten folgende eindeutigen Mappings aus Tabelle 9 auf.

Von den insgesamt 169 Mappings sind nun per Definition alle mit einem Wert über 0.5 false positives, abzüglich der 6 oben genannten true positives. Es gibt nun nur noch 15 (vorher 43) false positives. Bei diesen handelt es sich ausschließlich um Mappings zwischen Elementen, die nichts, „NULL“ oder unterschiedliche IDs enthalten.

Perseus	Epiduke	Ähnlichkeit
authors/dating	authors/dating	0.892923
authors/female	authors/female	1
works/corpora_author_id	works/corpora_author_id	0.91867995
works/work_id	works/work_id	0.71084154
sentences/s_id	sentences/s_id	0.80346525
sentences/sentence	sentences/sentence	0.89742255

Tabelle 9: Vergleichsergebnis von PHI7-DB versus Epiduke-DB.

DB-XML – Epiduke versus Epiduke

Gewichtet gibt Geminus die in Tabelle 10 gezeigten 1:1 Mappings.

Epiduke-DB	Epiduke-XML	Ähnlichkeit
authors/author	TEI/teiHeader/fileDesc/titleStmt/title	0.95708007
authors/dating	TEI/text/body/head/date	0.9740065
authors/geography	TEI/text/body/head/placeName	0.94999474
sentences/sentence	TEI/text/body	0.59451735

Tabelle 10: Vergleichsergebnis von Epiduke-DB versus Epiduke-XML.

Alle der 338 Ergebnisse mit einem Wert von mindestens 0.48700325 gelten als false positives. Abzüglich der 4 true positives sind das nun nur noch 9 (vorher 130). Von diesen handelt es sich bei dreien um annähernd äquivalente Textpaths, der Rest ist damit zu begründen, dass die „corpora_author_id“, die dem „author“ entspricht, auch in anderen Elementen als dem Titel auftritt, zum Beispiel in „publicationStmt“.

DB-XML – PHI7 versus Epiduke

Das berechnete eineindeutige Mapping ist nun wie in Tabelle 11.

PHI7-DB	Epiduke-XML	Ähnlichkeit
works/work	TEI/teiHeader/fileDesc/publicStmt/authority	0.17214279
sentences/sentence	TEI/text	0.32528445

Tabelle 11: Vergleichsergebnis von PHI7-DB versus Epiduke-XML.

Die Anzahl anderer Mappings mit einem Wert über 0.162642225 ist nun nur noch 5 (vorher 204), bei 338 insgesamt. Trotz des geringen Wertes erreichen nun nur noch sehr wenige Mappings dieses Niveau. Dabei handelt es sich wieder um 3 äquivalente Textpaths und ein Mapping von „sentences“ zum Wurzelknoten. Das unerwünschte eineindeutige false positive ist durch das häufige Auftreten des Wortes „inscriptions“ zu begründen, das nur zufällig und zusammenhangslos beide beinhalten.

DB-XML – MISC versus Epiduke

Das gewünschte Mapping wird nun eindeutig gefunden, wie in Tabelle 12 zu sehen, wenn auch mit einem geringen Wert.

MISC	Epiduke-XML	Ähnlichkeit
sentences/sentence	TEI/text	0.28001532

Tabelle 12: Vergleichsergebnis von MISC-DB versus Epiduke-XML.

Es gibt hier nur noch 4 weitere Mappings mit einem Wert über 0.14000766, und das sind wieder die 3 äquivalenten Textpaths und das Mapping von „sentences/sentence“ zu „TEI“.

XML-XML – Perseus versus Epiduke

Die gewichtet berechneten eindeutigen Mappings sind ebenfalls in den Anlagen (B) zu finden.

Unter den 1040 Mappings sind also 15 true positives. Von vorher 279 gelten nun nur 25 mit einem Wert über 0.429086 als false positives. Fast alle sind hierbei entweder Container-Elemente zu anderen oder äquivalente Paths. Lediglich das Mapping von „TEI.2/teiHeader/fileDesc“ zu „TEI/teiHeader/profileDesc“ mit 0.45410407 ist von denen auf diese Weise nicht zu begründen.

XML-XML – American History versus Richmond Times

Hier ändert sich annähernd nichts durch eine solche Gewichtung.

4.4 Wertung

Nach [5] ist eine der besten Möglichkeiten, etwas über die Ergebnisse von Schema-Matching-Verfahren auszusagen, die Gegenüberstellung von *Precision* und *Recall*. Diese berechnen sich beziehend auf die Mengen in Abbildung 4 auf Seite 21 wie folgt:

$$Precision = \frac{|TP|}{|TP| + |FP|} \quad (9)$$

$$Recall = \frac{|TP|}{|FN| + |TP|} \quad (10)$$

Die Precision gibt demnach Auskunft darüber, welche Anteile der gefundenen Mappings tatsächlich im Goldstandard liegen, während Recall reflektiert, welcher Anteil des Goldstandards mit den gefundenen Mappings abgedeckt wird. Als kombiniertes Maß dieser beiden Größen wird *F-Measure* benutzt.

$$F\text{-Measure}(\alpha) = \frac{\textit{Precision} \cdot \textit{Recall}}{(1 - \alpha) \cdot \textit{Precision} + \alpha \cdot \textit{Recall}} \quad (11)$$

Der Parameter α beträgt hier 0.5, da Precision und Recall gleich stark in den F-Measure-Wert eingehen sollen.

Für die eindeutigen Mappings aus Abschnitt 4.1 und 4.3 ergeben sich folgende Werte. Die Brüche bei Precision und Recall sind nicht gekürzt und ausgewertet, damit erkennbar bleibt, wie viele positives und negatives jeweils in der jeweiligen Berechnung beteiligt gewesen sind.

		1	2	3	4
ohne Gewichtung	Precision	6/6	4/5	1/3	14/17
	Recall	6/6	4/4	1/1	14/15
	F-Measure	1	0,8888	0,4962	0,8749
mit Gewichtung	Precision	6/6	4/4	1/2	15/15
	Recall	6/6	4/4	1/1	15/15
	F-Measure	1	1	0,6666	1

Tabelle 13: Precision, Recall und F-Measure für 1: Pers-DB vs Epi-DB, 2: Epi-DB vs Epi-XML, 3: PHI7-DB vs Epi-XML, 4: Pers-XML vs Epi-XML.

Anlage F visualisiert die Tabelle 13.

Bei zum Teil nur sehr wenigen Einträgen im Goldstandard sind diese hohen Werte mit Vorsicht und Skepsis zu betrachten. Sie bedeuten nicht zwangsläufig, dass Geminus ein ausgezeichnetes Programm ist, sondern lediglich, dass es auf den zum Evaluieren verwendeten Daten sehr gut funktioniert. Möglicherweise gilt: „Wenn die Ergebnisse zu gut sind, war das Problem zu einfach“¹⁰.

4.5 Weitere Parameter

Welche weiteren Einstellungsmöglichkeiten Einfluss auf das Ergebnis nehmen können, wird nun erläutert.

Als Stoppwörter werden hier diejenigen Wörter bezeichnet, die am häufigsten auftreten. Per Parameter lässt sich prozentual festlegen, wie viele Wörter entfernt, also bei der Berechnung der Ähnlichkeiten nicht berücksichtigt werden. Ob es auch sinnvoll ist, wenn die Wörter schon durch TF-IDF gewichtet sind, also häufige Wörter grundsätzlich schon herabgestuft sind, zeigt Abbildung 8. In ihr ist zu sehen, wie sich die Vergleichsergebnisse von Kosinus- und Dice-Ähnlichkeit ändern, wenn ein prozentualer Stoppwortfilter eingesetzt wird. Dem Vergleich liegt das Mapping „TEI/text/body“ mit „TEI.2/text/body“ der Perseus und Epiduke XML-Inputs zugrunde.

¹⁰Mdl. Kommentar von Prof. Dr. Martin Middendorf.

Das Entfernen von seltenen Wörtern, spezifisch solchen mit Frequenz 1, scheint im Gegensatz zur Stoppwort-Entfernung sinnvoll zu sein, da sich die Ergebnisse zum Teil erheblich verbessern.

Siehe dazu Abbildung 9. Ihr liegt dieselbe Berechnung zugrunde, wie in Abbildung 8, nur dass hier zusätzlich die seltensten Wörter entfernt wurden. Insbesondere die Dice-Ähnlichkeit scheint stark davon zu profitieren. Die Begründung dafür liegt auf der Hand. Kommt ein Wort nur einmal vor, so bedeutet das insbesondere, dass es nur in einem der Inputs auftritt. Wird dieses dann nicht berücksichtigt, so kann man es als eine künstliche Angleichung der Eingabeinformationen auffassen. Bei Dice wirkt es sich stärker aus, da hier die Frequenz irrelevant ist, sondern nur das pure Vorkommen des Wortes.

Ein weiterer Nachteil der Entfernung von seltenen Wörtern ist möglicherweise Informationen zu ignorieren, die im Header stehen und auf natürliche Weise dort nur einmal vorkommen. Diese Entscheidung bleibt dem Anwender überlassen.

Das Ignorieren von Nummern ist bereits in Abschnitt 2.2 motiviert worden. Dazu wird hier eine Teilmenge der Perseus- und Epiduke-XML-Dateien verglichen. Wieder jeweils zwischen den Paths „text/body“ ergaben sich Werte wie in Tabelle 14.

Es entstehen hier also keine nennenswerten Unterschiede. Interessanterweise verschlechtert sich der Dice-Wert sogar etwas, während die Kosinusähnlichkeit unwesentlich davon profitiert.

ignore numbers	cosine	dice
false	0.8865594	0.76700515
true	0.8919656	0.7627859

Tabelle 14: Änderungen der Werte durch das Ignorieren von Nummern.

Welchen Einfluss hat nun die Betacode-Konversion auf die Korpora? Dazu wird die Perseus-Datenbank mit den Epiduke-XML-Dateien verglichen, welche beide in UTF-8 Griechisch vorliegen. Die Wörter werden nicht normalisiert. In einem zweiten Durchlauf werden dann beide Inputs in Betacode überführt und erneut verglichen. Betrachtet wird

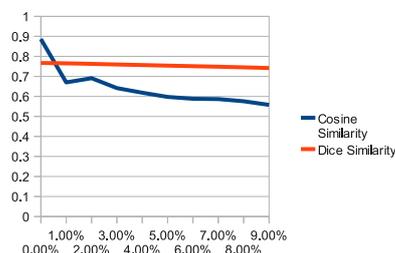


Abbildung 8: Die Ähnlichkeit nimmt ab, je mehr Stoppwörter entfernt werden.

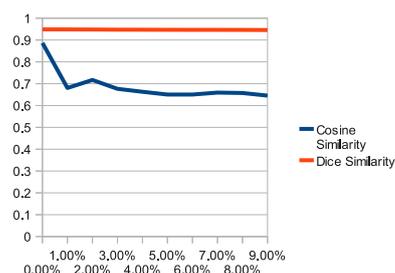


Abbildung 9: Die Ähnlichkeit nimmt auch ohne seltene Wörter ab, allerdings sind die Ergebnisse insgesamt besser.

in Tabelle 15 das Mapping von „Perseus/sentences/sentence“ mit „TEI/text/body“.

encoding	cosine	dice
unicode	0.57096565	0.35732266
betacode	0.5992725	0.47221008

Tabelle 15: Änderungen der Werte durch unterschiedliche Kodierung des Griechischen.

Für diesen Unterschied, insbesondere bei der Dice-Ähnlichkeit, gibt es eine einfache Begründung. Einzelne Zeichen sind anders kodiert; zum Beispiel kann ein α aus dem „Greek and Coptic“-Unicodeblock kommen, oder aber aus dem „Extended Greek“-Block. Durch die Konversion in lateinische Buchstaben wird die Kodierung wieder einheitlich.

Die Normalisierung umgeht das Problem einer nicht einheitlichen Kodierung definitiv und erreicht in Tabelle 16 sichtbare Verbesserungen der Werte (bei dem Unicode-Input wie zuvor).

normalize	cosine	dice
false	0.57096565	0.35732266
true	0.56675494	0.48062035

Tabelle 16: Änderungen der Werte durch Normalisierung.

Die Dice-Ähnlichkeit steigt erwartungsgemäß ähnlich wie zuvor an, allerdings verschlechtert sich die Kosinusähnlichkeit kaum merklich durch die Angleichung der Wörter.

Es scheint also eher weniger ratsam, Wörter oder Zahlen aus den Inputs zu ignorieren beziehungsweise zu streichen. Das Ändern der Kodierung kann allerdings – insbesondere bei der dice similarity – zu deutlich besseren Ergebnissen führen.

Zum Abschluss der Arbeit wird nun auf Erfahrungen und weitere mögliche Arbeiten eingegangen, bevor die Ergebnisse kurz zusammengefasst werden.

5 Lessons Learnt und Further Work

Die Entwicklung von Geminus und das Schreiben dieser Arbeit ermöglichte eine umfangreiche Erkenntnisgewinnung. Ein Großteil davon ist die Sammlung von Erfahrungen, nur ein Bruchteil davon lässt sich auch in Worte fassen.

Was wurde erreicht und was nicht? Geminus ist in der Lage, auch umfangreiche Datensammlungen innerhalb weniger Minuten zu verarbeiten. Die Kollektionen können sich stark ähneln, in inhaltlichen oder strukturellen Aspekten.

Hierbei benötigt das Auslesen von Datenbanken bedeutend weniger Zeit, als die Verarbeitung von XML-Dateien. Die Begründung dafür liegt in dem zusätzlichen Aufwand, für jede XML-Datei einen DOM-Baum zu erstellen. Ferner werden aufgrund der verschachtelten Paths fast alle Textpassagen mehrfach verarbeitet, je nach Pfadtiefe (das heißt „TEI/text/body/div“ enthält eine Teilmenge des Textes von „TEI/text/body“, welcher eine Teilmenge von „TEI/text“ enthält, und so weiter). Dabei wurde schnell deutlich, dass die instanzbasierten Verfahren schnell zu Komplexitätsproblemen führen können, sowohl in Bezug auf die Zeit, als auch auf den Speicher. Um die „Term-Dokument“-Matrix mit $|Words| \times |Paths|$ Elementen im Speicher also klein zu halten, ergaben sich nur zwei Möglichkeiten. Die Anzahl der Wörter wurde durch Tokenisierung, Normalisierung, Ignorieren von Ziffern und Entfernen von sehr häufigen und sehr seltenen Wörtern in Grenzen gehalten, wobei nicht jedes dieser Optionen im Nachhinein tatsächlich Sinn macht. Wesentlich wichtiger war die Eindämmung der Zahl der (XML-)Paths, einhergehend mit der Erkenntnis, dass bei weitem nicht jeder Path auch in jeder Datei vorkam. Da es auch nicht zielführend ist, für einen Path ein Mapping zu generieren, der nur in einem Bruchteil der Dokumentsammlung auftritt, können so – abhängig vom Input – zum Teil über 90% der gefundenen Paths ignoriert werden.

Am Ende des Algorithmus steht eine übersichtliche Veranschaulichung der Vergleichsergebnisse, die durch Mouseover über die einzelnen Zellen eine gewisse Möglichkeit der Analyse bietet. Hat ein Mapping beispielsweise nur einen mittelmäßigen Wert, so lässt sich leicht herausfinden, welche Verfahren dafür verantwortlich sind. Ist die Namensähnlichkeit hoch, aber die Kosinus- und die Dice-Ähnlichkeit gering, so könnte es sich um ein strukturell gleiches Element handeln, obwohl der Inhalt nicht zusammen passt. Ist es andersherum, so könnte es ein Element mit der gleichen semantischen Bedeutung sein, obwohl es einen anderen Namen trägt. In Kombination mit den anderen Verfahren im Blickfeld lässt sich die generelle Übereinstimmung zwischen den beiden Inputs so innerhalb weniger Sekunden erfassen, das heißt, ob die Deckung nur strukturell bedingt ist oder ob inhaltliche Ähnlichkeiten bestehen.

Als zukünftig einsetzbar könnten sich auf jeden Fall die Dice- und die Kosinusähnlichkeit erweisen. Diese lieferten annähernd konstant gute und brauchbare Ergebnisse, mit Ausnahme von Vergleichen, die beabsichtigt unterschiedliche Inhalte hatten und ein Mapping allein aufgrund der Strukturinformationen erzeugen sollten. Keines der anderen Verfahren lieferte für sich alleine eine solch genaue Deckung mit den true positives. Auf die Inhalte der Paths zu schauen zieht natürlich enorme Performanznachteile mit sich, die Präzision

ist diese allerdings oft wert.

Die Namensähnlichkeit erzeugte auch zum Großteil sinnvolle Ergebnisse, insbesondere, wenn die Inputs auf dem gleichen Schema basierten, egal ob Datenbank oder XML.

Ein interessantes Ergebnis ist die hohe Ähnlichkeit der englischen Korpora. Obwohl diese inhaltlich eher wenige Gemeinsamkeiten haben, ergaben die instanzbasierten Vergleiche das Gegenteil. Eine mögliche Untersuchung wäre nun, ob es an der großen Textmasse liegt, in Verbindung mit der Tatsache, dass das Englische einen geringeren Wortschatz hat als das Altgriechische.

Es gibt auch eine Vielzahl an weiteren Punkten, die zukünftig als weitere Arbeitsschritte betrachtet werden können, die allerdings den Rahmen dieser Bachelorarbeit überschreiten würden.

Ein Problem bleibt nach wie vor die Gewichtung. Prinzipiell müsste man im Voraus in die Daten sehen, um festzustellen, ob es tendenziell strukturelle Ähnlichkeiten gibt oder nicht, und wie stark man dem entsprechend die dafür zuständigen Verfahren einfließen lässt. Ein denkbare Experiment wäre, die Gewichte automatisch zu setzen. Basierend auf der Verteilung der einzelnen erreichten Werte könnte ein Verfahren, welches durchgehend sehr gute Werte verteilt oder aber bei keinem Mapping auch nur annähernd gute Ergebnisse erzielt, gering gewichtet werden. Im Gegensatz dazu würden die Verfahren mit hohen Gewichten belohnt, die bei wenigen Mappings hohe Ergebnisse erzielen, und dann stark abfallen. Es müsste dann empirisch nachgewiesen werden, dass diese wenigen, hohen Vergleichswerte tatsächlich auch die true positives sind.

Weiterhin denkbar wäre eine klare Trennung von inhaltlichen und strukturellen Methoden, einhergehend mit einer Priorisierung. So könnte zum Beispiel zunächst versucht werden, inhaltliche Übereinstimmungen zu finden, und wenn das nicht gelingt, auf Strukturähnlichkeiten zurückzugreifen, die Gewichte also während der Laufzeit zu ändern beziehungsweise einen neuen Berechnungsschritt zu starten.

Geminus ist in der Lage, mit lateinischen und griechischen Buchstaben zu arbeiten. Grundsätzlich wäre es kein Problem, auch anderssprachige Inputs zu verarbeiten, allerdings besteht die Möglichkeit, dass auch hebräische, asiatische oder russische Schriften eine Kodierung in lateinische Buchstaben ähnlich des BetaCode besitzen. In diesem Fall wäre es denkbar diese Kodierungen zu unterstützen, um so unterschiedlichere Inputs und ihre Kodierungen verarbeiten zu können.

Die Auswertung der Formate, die sich als Input eignen, können ebenfalls erweitert werden. Durch das Auswerten von DTDs können zusätzliche strukturelle Informationen gewonnen werden. Die relationalen oder objektorientierten Datenbankschemata können analog dazu ebenfalls umfassender ausgewertet werden. Das Erfassen von XML-Attributen und nicht nur der Tags als Paths verbindet möglicherweise ebenfalls sinnvolle Mapping-Partner miteinander, die momentan unberücksichtigt bleiben. Bei der bisherigen Betrachtung und den verwendeten Daten, insbesondere bei dem Vergleich von XML-Dateien mit Datenbanken, schien es nicht besonders sinnvolle Übereinstimmungen zu geben, weshalb dieser Teil der XML-Dateien nicht ausgewertet wurde.

Neben XML und Datenbanken sind weitere Formate denkbar (komplexere relationale Schemata, eventuell auch Modelle wie RDF oder OWL), sogar proprietäre, solange sich daraus Paths identifizieren lassen, die Informationen wie Texte beinhalten.

Die Verfahren, die verwendet werden, lassen sich zum Teil verfeinern oder ersetzen.

Für eine Spezialisierung auf Textkorpora als Inputs ließen sich Verfahren entwickeln, bei denen die Sicherheit für eine Ähnlichkeit steigt, je länger aufeinander folgende und übereinstimmende Abschnitte gefunden werden. Bisher wurden Wörter nur als Mengen und Paths als ungeordnet betrachtet, aber der Linearität des Textes kann durchaus mehr Aufmerksamkeit geschenkt werden. Mit einem solchen Ansatz ließen sich auch besser Teilmengen und -abschnitte von Texten identifizieren, als durch einen Vektorvergleich nur eine Zahl wie „0,58“ zu erhalten.

Obwohl im Kosinus-Vektorvergleich indirekt die Termfrequenzen enthalten sind, ist ein zusätzliches Maß interessant und möglicherweise brauchbar, welches auf die in den Paths enthaltenen Textmengen eingeht. Es müsste insbesondere untersucht werden, ob die Kosinusähnlichkeit einen hohen Wert bekommen kann, obwohl die Paths stark unterschiedlich große Textmengen enthalten. Damit einhergehend muss überlegt werden, ob dieser Vergleich nicht nachteilig ist, wenn die vorliegenden Daten nur Teilabschnitte voneinander sind.

Die TF-IDF-Gewichtung der Wörter im Vektor lässt sich auch verfeinern. Da hier die Paths als Dokumente betrachtet werden, entstehen Nebenwirkungen. Wie bereits genannt, werden einige Terme aus den XML-Inputs mehrfach durchlaufen, je nach Hierarchieebene. Das bedeutet insbesondere, dass diese Wörter auch in mehreren Paths vorkommen, je nachdem, wie tief sie liegen. Gerade dadurch allerdings werden sie durch die inverse Dokumentfrequenz unnötig abgewertet. Wie sich in der Evaluation gezeigt hat, wirkt es sich nicht zu stark negativ aus, lässt allerdings Spielraum zur Verbesserung.

Außerdem ist bereits im Beispiel in Abschnitt 3.2.3 erkennbar, dass eine solche Gewichtung auf sehr geringen Daten zu keinen zufriedenstellenden Ergebnissen führt. In eine Lösung dieses Problems kann ebenfalls Zeit investiert werden, und zwar nicht nur für diesen konkreten Anwendungsfall.

Abschließend lassen sich noch viele weitere Tests durchführen, um die Funktion von Geminus zu bestätigen. Es lagen zum Zeitpunkt der Arbeit keine Daten vor, die ungeordnet und strukturell unterschiedlich, aber semantisch sehr ähnlich waren, wie beispielsweise Personaldaten oder Ähnliches. Es scheint auch kein Benchmark zu existieren, der die Zuverlässigkeit von Schema-Matching-Algorithmen allgemeingültig testet.

Es ist also durchaus denkbar, noch weitere Arbeiten folgen zu lassen.

6 Zusammenfassung

Das wesentliche Ziel der Arbeit bestand darin, strukturelle oder inhaltliche Übereinstimmungen in verschiedenen Textkorpora und deren Metadaten zu finden. Es wurden zu diesem Zweck die Ähnlichkeiten zwischen verschiedenen Korpora mit unterschiedlichen Datenformaten und -quellen untersucht, unter Zuhilfenahme eines eigens dafür entwickelten Programms. Die Daten lagen als XML-Dateien und Datenbanken vor und beinhalten hauptsächlich griechische Sprache.

Um mit dem griechischen Zeichensatz und der teilweise ebenfalls vorhandenen Kodierung in Betacode umzugehen, wurde eine optionale Konversion von Unicode nach Betacode eingesetzt. Um unterschiedlichen Unicode-Kodierungen vorzubeugen, wurde eine Grundformreduktion auf Ebene der Buchstaben verwendet, hier Normalisierung genannt.

Nachdem die Inputs eingelesen und in atomare Vergleichseinheiten – sogenannte Paths – unterteilt wurden, erzeugten verschiedene Verfahren Ähnlichkeitswerte, basierend auf unterschiedlichen Aspekten der Paths. Als besonders hilfreich erwiesen sich die instanzbasierten Verfahren, welche direkt Inhalte von Paths miteinander verglichen. Sie konnten zuverlässig übereinstimmende Texte identifizieren, sofern die Inhalte der Inputs tatsächlich übereinstimmten, wenn auch nur in Teilen.

Dass die Gewichtung der einzelnen Verfahren in Abhängigkeit von der Form des Inputs die Qualität des Gesamtergebnisses wesentlich beeinflusst, wurde im Zuge der Evaluation sehr deutlich. Ebenso deutlich wurde die Wichtigkeit weiterer Verarbeitungsschritte, um unnütze Informationen zu verwerfen, damit sowohl die Performanz als auch die Qualität gesteigert werden konnten.

Der manuell gesetzte Goldstandard wurde nahezu immer vollständig gefunden und die zusätzlichen ‘falschen Ähnlichkeiten’ hielten sich – insbesondere bei sinnvoller Gewichtung – in Grenzen.

Durch die zusätzliche Ausgabe einer kolorierten Tabelle und eindeutiger Struktur mappings arbeitet das Programm sehr sauber und nachvollziehbar. Auch der Vergleich von Daten, die nicht aus Textinformationen bestehen, ist so im Nachhinein denkbar.

Zusammenfassend lässt sich sagen, dass das entwickelte Programm zuverlässig arbeitet, auch wenn sich nicht alle Verfahren als unbedingt sinnvoll herausstellen und andere wiederum die Möglichkeit offen lassen, zukünftig weitere Arbeitsschritte anzuknüpfen. Das Ziel, durch ein Schema-Matching-Algorithmus Arbeit und Zeit zu sparen, sowie eine Datenintegration effizient vorzubereiten, wurde erreicht.

7 Literaturverzeichnis

- [1] ALGERGAWY, A.; NAYAK, R.; SAAKE, G. (2009): „XML Schema Element Similarity Measures: A Schema Matching Context“, *Proceedings of OTM*.
- [2] BAEZA-YATES, R.; RIBEIRO-NETO, B. (1999): *Modern Information Retrieval*, New York, 27-30.
- [3] BERLIN, J.; MOTRO, A. (2002): „Database Schema Matching Using Machine Learning with Feature Selection“, *CAISE 2002, LNCS 2348*, 452–466.
- [4] COHEN, W.; RAVIKUMAR, P.; FIENBERG, S. (2003): „A comparison of string metrics for matching names and records“, *Proceedings of the workshop on Data Cleaning and Object Consolidation at the International Conference on Knowledge Discovery and Data Mining (KDD)*.
- [5] DO, H.H.; MELNIK, S.; RAHM, E. (2002): „Comparison of Schema Matching Evaluations“, *Proc. GI-Workshop ‘Web and Databases’*, Erfurt.
- [6] DO, H.H.; RAHM, E. (2002): „COMA - A system for flexible combination of Schema Matching Approaches“, *Very Large Databases (VLDB)*.
- [7] DOAN, A.; DOMINGOS, P.; HALEVY, A. (2001): Reconciling schemas of disparate data sources: a machine-learning approach, *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, Santa Barbara, California, United States, 509-520.
- [8] DOAN, A.; HALEVY, A. (2005): „Semantic Integration Research in the Database Community: A Brief Survey“, *AI Magazine, Special Issue on Semantic Integration*.
- [9] GIUNCHIGLIA, F.; YATSKEVICH, M. (2004): „Element level semantic matching“, *Proceedings of Meaning Coordination and Negotiation workshop at the International Semantic Web Conference (ISWC)*.
- [10] HALEVY, A. (2005): „Why Your Data Don’t Mix“, *ACM Queue*.
- [11] HAN, J.; KAMBER, M. (2006): *Data Mining: Concepts and Techniques*, Second Edition, 67-71; 614-627.
- [12] HEYER, G.; QUASTHOFF, U.; WITTIG, T. (2006): *Text Mining: Wissensrohstoff Text*, 201-209.
- [13] KONDRAK, G. (2005): „N-Gram Similarity and Distance“, *SPIRE 2005, LNCS 3772*, 115–126.
- [14] MADHAVAN, J.; BERNSTEIN, P.A.; RAHM, E. (2001): „Generic Schema Matching with Cupid“, *Proc. 27th Intl. Conference on Very Large Databases (VLDB)*, Rome, Italy.
- [15] MARZAL, A.; VIDAL, E. (1993): „Computation of normalized edit distance and applications“, *IEEE Trans. Pattern Analysis and Machine Intelligence 15(9)*, 926–932.

-
- [16] RAHM, E.; BERNSTEIN, P.A. (2001): „A survey of approaches to automatic schema matching“, *VLDB Journal* 10(4).
- [17] SALTON, G.; BUCKLEY, C. (1988): „Term-weighting approaches in automatic text retrieval“, *Information Processing and Management*.
- [18] SHVAIKO, P; EUZENAT, J. (2005): „A Survey of Schema-based Matching Approaches“, *Journal on Data Semantics*.
- [19] THANG, H.Q.; NAM, V.S. (2010): „XML Schema Automatic Matching Solution“, *International Journal of Electrical, Computer, and Systems Engineering*.
- [20] XU, L.; EMBLEY, D.W. (2003): „Using domain ontologies to discover direct and indirect matches for schema elements“, *Proceedings of the Semantic Integration workshop at the International Semantic Web Conference (ISWC)*.
- [21] kcl.ac.uk (Stand Mai 2010): *Epiduke vom King's College London*.
- [22] www.eaqua.net (Stand Juni 2010): *Extraktion von strukturiertem Wissen aus Antiken Quellen für die Altertumswissenschaft*.
- [23] www.eaqua.net/~dpansch/tool_documentation/geminus.php (Stand Juni 2010): *Geminus Documentation*.
- [24] www.perseus.tufts.edu/hopper/ (Stand Juni 2010): *Perseus Digital Library*.
- [25] www.tei-c.org (Stand Juni 2010): *Text Encoding Initiative*.
- [26] www.timesdispatch.com (Stand Juni 2010): *Richmond Times-Dispatch*.

8 Anlagenverzeichnis

Anlage A auf Seite 42: Die eindeutigen Mappings von Perseus-XML und Epiduke-XML aus Abschnitt 4.1, die aufgrund ihrer Größe keinen Platz im Text gefunden haben.

Anlage B auf Seite 43: Die eindeutigen Mappings von Perseus-XML und Epiduke-XML aus Abschnitt 4.3.

Anlage C auf Seite 44: Ein Screenshot des JTable-Outputs, der am Ende des Prozesses steht. Hier ist der Vergleich von Perseus-XML mit Epiduke-XML zu sehen. In einer übersichtlichen Matrix werden die einzelnen Mappings angeordnet und je nach Wert mehr oder weniger stark grün eingefärbt. Ein Tooltip bei Mouseover zeigt weitere Informationen zu den einzelnen Zellen an, wie rechts zu sehen.

Anlage D auf Seite 45: Eine Beispielausgabe des Vergleichs von Perseus-XML mit Epiduke-XML bis zu einer Ähnlichkeit von 0,5. Die ursprünglich tabseparierte CSV-Datei wird hier tabellarisch dargestellt. Die initialen laufenden Nummern sind nicht Bestandteil des Outputs, sondern dienen hier nur der Zeilenzuordnung, da die Tabelle aufgrund ihrer Breite in zwei Teile aufgeteilt wurde.

Anlage E auf Seite 46: Die Datei o.camb.26.xml als beispielhafter Auszug aus der Epiduke-XML-Sammlung. Wie man leicht sieht, muss der eigentliche Textinhalt der Dateien nicht groß ausfallen.

Anlage F auf Seite 47: Die Visualisierung der Tabelle 13 aus Abschnitt 4.4. Sie stellt die Verbesserung der Precision-, Recall- und F-Measure-Werte durch die Gewichtung der verwendeten Verfahren dar.

Perseus	Epiduke	Ähnlichkeit
TEI.2	TEI	0.6395043
TEI.2/teiHeader	TEI/teiHeader	0.5917362
TEI.2/teiHeader/fileDesc	TEI/teiHeader/fileDesc	0.56577283
TEI.2/teiHeader/fileDesc/titleStmt	TEI/teiHeader/fileDesc/titleStmt	0.57119346
<i>TEI.2/teiHeader/fileDesc/titleStmt/respStmt/resp</i>	<i>TEI/teiHeader/fileDesc/publicationStmt/availability/p</i>	<i>0.45438182</i>
TEI.2/teiHeader/fileDesc/publicationStmt	TEI/teiHeader/fileDesc/publicationStmt	0.5610782
TEI.2/teiHeader/fileDesc/sourceDesc	TEI/teiHeader/fileDesc/sourceDesc	0.5716683
<i>TEI.2/teiHeader/fileDesc/sourceDesc/bibl</i>	<i>TEI/teiHeader/fileDesc/titleStmt/title</i>	<i>0.48586833</i>
TEI.2/teiHeader/profileDesc	TEI/teiHeader/profileDesc	0.77592427
TEI.2/teiHeader/profileDesc/langUsage	TEI/teiHeader/profileDesc/langUsage	0.77592427
TEI.2/teiHeader/profileDesc/langUsage/language	TEI/teiHeader/profileDesc/langUsage/language	0.684027
TEI.2/text	TEI/text	0.7701943
TEI.2/text/body	TEI/text/body	0.7701943
TEI.2/text/body/div1/head	TEI/text/body/head	0.6167437
TEI.2/text/body/div1/head/date	TEI/text/body/head/date	0.6047558
TEI.2/text/body/div1/head/placeName	TEI/text/body/head/placeName	0.62667376
<i>TEI.2/text/body/div1/p/lb</i>	<i>TEI/text/body/div/ab/lb</i>	<i>0.4901842</i>

Anlage A: Die eindeutigen Vergleichsergebnisse von Perseus- und Epiduke-XML ohne Gewichte.

Perseus	Epiduke	Ähnlichkeit
TEI.2	TEI	0.6486199
TEI.2/teiHeader	TEI/teiHeader	0.5390482
TEI.2/teiHeader/fileDesc	TEI/teiHeader/fileDesc	0.49986154
TEI.2/teiHeader/fileDesc/titleStmt	TEI/teiHeader/fileDesc/titleStmt	0.49966556
TEI.2/teiHeader/fileDesc/titleStmt/title	TEI/teiHeader/fileDesc/titleStmt/title	0.43290916
TEI.2/teiHeader/fileDesc/publicationStmt	TEI/teiHeader/fileDesc/publicationStmt	0.49254215
TEI.2/teiHeader/fileDesc/sourceDesc	TEI/teiHeader/fileDesc/sourceDesc	0.5
TEI.2/teiHeader/profileDesc	TEI/teiHeader/profileDesc	0.8570907
TEI.2/teiHeader/profileDesc/langUsage	TEI/teiHeader/profileDesc/langUsage	0.8570907
TEI.2/teiHeader/profileDesc/langUsage/language	TEI/teiHeader/profileDesc/langUsage/language	0.79312587
TEI.2/text	TEI/text	0.858172
TEI.2/text/body	TEI/text/body	0.858172
TEI.2/text/body/div1/head	TEI/text/body/head	0.69373125
TEI.2/text/body/div1/head/date	TEI/text/body/head/date	0.6687381
TEI.2/text/body/div1/head/placeName	TEI/text/body/head/placeName	0.7046045

Anlage B: Die eindeutigen Vergleichsergebnisse von Perseus- und Epiduke-XML mit Gewichten.

Geminus - Comparison Results																
	TEI	teiHeader	fileDesc	titleStmnt	title	profileDesc	langUsage	language	revisionD...	change	text	body	head	date	placeName	div
TEI.2	0.56505	0.19692	0.18192	0.16974	0.16474	0.17907	0.17074	0.07747	0.17807	0.10249	0.56083	0.54416	0.23313	0.21519	0.19953	0.52712
teiHeader	0.20648	0.41565	0.24703	0.23421	0.24088	0.29561	0.26834	0.18674	0.25504	0.15029	0.28354	0.23354	0.25088	0.22421	0.22421	0.21686
fileDesc	0.18298	0.24931	0.39703	0.27588	0.24838	0.35997	0.22588	0.12261	0.31338	0.17738	0.23563	0.25021	0.24463	0.22963	0.22754	0.24394
titleStmnt	0.17464	0.23264	0.27203	0.40088	0.31421	0.25315	0.25088	0.14261	0.23838	0.18363	0.25021	0.22521	0.26754	0.26421	0.26421	0.26686
title	0.10473	0.17439	0.17962	0.24930	0.33596	0.17324	0.18263	0.20658	0.17096	0.25588	0.15530	0.14530	0.16596	0.24596	0.20263	0.19528
respStmnt	0.16964	0.20597	0.20703	0.29754	0.26963	0.22451	0.23088	0.16261	0.24838	0.16363	0.22771	0.21021	0.24963	0.26963	0.26754	0.23019
resp	0.16631	0.19931	0.23453	0.23421	0.23421	0.24179	0.23421	0.14594	0.23838	0.15029	0.22104	0.20021	0.25504	0.23421	0.25088	0.21686
name	0.16631	0.21597	0.21578	0.23421	0.26421	0.21451	0.26754	0.20219	0.21338	0.20029	0.18354	0.20021	0.21754	0.30921	0.28421	0.21686
publicati...	0.17464	0.20931	0.23203	0.31088	0.26088	0.24588	0.27088	0.16261	0.26588	0.20363	0.22021	0.24521	0.26088	0.25088	0.27088	0.26019
distributor	0.16964	0.21658	0.22067	0.27179	0.29179	0.21088	0.23088	0.17625	0.22338	0.16363	0.21748	0.22385	0.23088	0.27815	0.25088	0.25747
sourceDe...	0.17464	0.21431	0.29703	0.26588	0.24588	0.30770	0.25088	0.15761	0.27588	0.19863	0.21521	0.24021	0.26588	0.24588	0.26088	0.25019
bibl	0.16964	0.20597	0.22578	0.26421	0.31088	0.22451	0.23088	0.16261	0.22338	0.16363	0.19021	0.24771	0.23088	0.25088	0.25088	0.26769
encoding...	0.18298	0.22847	0.30953	0.22588	0.23588	0.32588	0.25088	0.14761	0.31338	0.20863	0.22938	0.27521	0.23838	0.23588	0.22338	0.25019
refsDecl	0.17464	0.21597	0.25953	0.25088	0.24963	0.28042	0.25088	0.14261	0.28838	0.20238	0.21896	0.22521	0.26963	0.24963	0.23088	0.25019
state	0.16964	0.22264	0.22578	0.26421	0.31088	0.22451	0.26421	0.20011	0.23588	0.21363	0.22021	0.21021	0.26088	0.34088	0.28421	0.23019
profileDesc	0.21317	0.28674	0.35612	0.25315	0.23815	0.59941	0.43804	0.33478	0.31338	0.17226	0.23051	0.26385	0.23951	0.22451	0.23815	0.23883
langUsage	0.20483	0.25947	0.22203	0.25088	0.24754	0.43804	0.59941	0.45781	0.23838	0.23363	0.20021	0.22521	0.26754	0.26421	0.28088	0.25019
language	0.13492	0.20122	0.14212	0.16596	0.20471	0.35813	0.48116	0.53636	0.15846	0.28713	0.12530	0.14530	0.18471	0.22346	0.25263	0.16528
handList	0.17464	0.19931	0.24078	0.26754	0.23088	0.23951	0.28421	0.18011	0.25088	0.22113	0.21896	0.24396	0.28838	0.24963	0.24754	0.28769
hand	0.09131	0.12764	0.12870	0.15254	0.17254	0.13254	0.18588	0.22969	0.14504	0.22585	0.11188	0.13188	0.22754	0.21004	0.18921	0.15186
text	0.42554	0.27961	0.23397	0.24471	0.21471	0.22501	0.19471	0.09644	0.22388	0.12746	0.76177	0.57844	0.29571	0.23525	0.23620	0.55305
body	0.40888	0.22961	0.24855	0.21971	0.20471	0.25835	0.21971	0.11644	0.25721	0.15246	0.57844	0.76177	0.28321	0.25525	0.23954	0.57805
milestone	0.17464	0.21597	0.27203	0.30088	0.28088	0.25315	0.26754	0.15928	0.27588	0.20029	0.23354	0.24188	0.26754	0.26421	0.24754	0.26686
div1	0.40055	0.21295	0.24231	0.26138	0.25471	0.23335	0.24471	0.14335	0.23221	0.17980	0.56178	0.58678	0.30822	0.31275	0.25954	0.71556
head	0.18709	0.23931	0.22578	0.24754	0.25088	0.22451	0.24754	0.18827	0.22338	0.19096	0.27249	0.25499	0.86869	0.65403	0.52930	0.23019
date	0.17995	0.21597	0.21578	0.25088	0.29421	0.21451	0.25088	0.19035	0.21338	0.20263	[TEI.2/text/body/div1]					
placeName	0.17422	0.21597	0.21370	0.25088	0.25088	0.22815	0.26754	0.21952	0.21338	0.18596	[TEI.2/text/body/div]					
p	0.39176	0.18609	0.20836	0.22452	0.24452	0.21815	0.22452	0.16316	0.20452	0.15961	NameSim: 0.75, PathSim: 1.0, VectorSim: 0.71200913, DiceSim: 0.71863276,					
lb	0.08200	0.09833	0.13147	0.14990	0.17990	0.13021	0.14990	0.22933	0.11657	0.16638	(Abs)FregSim: 1.0, (Rel)FregSim: 0.00877193, ContentTypeSim: 0.89043057					
expan	0.18940	0.12621	0.12636	0.14131	0.14131	0.12161	0.14131	0.22805	0.13297	0.16907	0.32084	0.30751	0.15464	0.14131	0.15797	0.32662
num	0.04758	0.06458	0.08353	0.11301	0.11301	0.07968	0.11301	0.16868	0.09218	0.17800	0.06368	0.08035	0.09635	0.11301	0.12968	0.09703
gap	0.07983	0.11282	0.11055	0.13106	0.14773	0.11439	0.16439	0.25944	0.11439	0.18485	0.09706	0.11373	0.16856	0.18523	0.16439	0.13037
num	0.05556	0.07019	0.08675	0.11385	0.11147	0.08290	0.11385	0.13225	0.09540	0.10169	0.06928	0.08357	0.09719	0.11147	0.12814	0.09787
app	0.14730	0.14429	0.14201	0.16253	0.17919	0.14586	0.17919	0.17357	0.14586	0.16229	0.18411	0.20077	0.16253	0.21669	0.19586	0.21816
lem	0.13064	0.13953	0.17237	0.20300	0.19729	0.16599	0.16967	0.16166	0.15122	0.12776	0.19584	0.17263	0.19050	0.16729	0.21729	0.18736
rdg	0.12911	0.14036	0.13571	0.15384	0.16813	0.15319	0.17051	0.16146	0.15205	0.15294	0.15889	0.21068	0.15384	0.16813	0.16813	0.18790
handShift	0.08444	0.10077	0.11516	0.15234	0.15234	0.11901	0.16901	0.25654	0.11901	0.18596	0.11834	0.13501	0.16901	0.16901	0.15234	0.16832
milestone	0.08429	0.11729	0.16501	0.18553	0.20219	0.14613	0.15219	0.21798	0.16886	0.16184	0.13486	0.13486	0.15219	0.18553	0.16886	0.15151
foreign	0.17865	0.19931	0.23453	0.23421	0.23421	0.22815	0.23421	0.16469	0.22588	0.17172	0.24920	0.26587	0.23897	0.25564	0.25088	0.28302
gap	0.08978	0.11860	0.11216	0.12851	0.14101	0.11601	0.16184	0.20044	0.11601	0.20117	0.10284	0.11534	0.16601	0.17851	0.15768	0.12782
corr	0.08978	0.11860	0.11216	0.12851	0.14101	0.12965	0.12851	0.16294	0.12851	0.20117	0.10284	0.15284	0.12851	0.14101	0.15768	0.12782
hi	0.10918	0.11603	0.13706	0.14779	0.17779	0.12810	0.13113	0.21866	0.12696	0.17309	0.18234	0.19901	0.16863	0.14779	0.14779	0.26667
expan	0.10296	0.13226	0.12969	0.14574	0.14336	0.12842	0.14574	0.21584	0.13979	0.18390	0.19086	0.17514	0.15907	0.14336	0.16003	0.19011
num	0.06727	0.08190	0.09846	0.12556	0.12318	0.09461	0.12556	0.13415	0.10711	0.14939	0.08099	0.09528	0.10890	0.12318	0.13985	0.10958
expan	0.08708	0.12869	0.12433	0.13860	0.13443	0.12307	0.13860	0.20691	0.13443	0.17675	0.13470	0.11720	0.15193	0.13443	0.15110	0.12979
expan	0.09411	0.13115	0.12679	0.14105	0.13689	0.12552	0.14105	0.19430	0.13689	0.18646	0.15678	0.13928	0.15439	0.13689	0.15355	0.15209
foreign	0.17309	0.18820	0.21786	0.21199	0.20643	0.21148	0.21199	0.13692	0.20921	0.14950	0.23809	0.24920	0.21675	0.22786	0.22310	0.26080
gap	0.08744	0.11805	0.11340	0.13153	0.14581	0.11724	0.16486	0.22198	0.11724	0.19360	0.10229	0.11658	0.16903	0.18331	0.16248	0.13084
note	0.11719	0.15018	0.16666	0.20175	0.24509	0.17903	0.20175	0.19605	0.17675	0.20892	0.13442	0.18859	0.16842	0.26009	0.20175	0.16774
space	0.10140	0.15106	0.15087	0.16930	0.19930	0.14960	0.18596	0.20866	0.16096	0.26754	0.11863	0.13530	0.18263	0.22930	0.21930	0.15195

Anlage C: Die Vergleichsergebnisse von Perseus- und Epiduke-XML als JTable.

Nr.	Input1	Input2	Score	Name Sim	Name Wght	Path Sim	Path Wght
1	"TEI.2/text"	"TEI/text"	0.858172	1	0.2	1	0.1
2	"TEI.2/text/body"	"TEI/text/body"	0.858172	1	0.2	1	0.1
3	"TEI.2/teiHeader/profileDesc"	"TEI/teiHeader/profileDesc"	0.8570907	1	0.2	1	0.1
4	"TEI.2/teiHeader/profileDesc/langUsage"	"TEI/teiHeader/profileDesc/langUsage"	0.8570907	1	0.2	1	0.1
5	"TEI.2/teiHeader/profileDesc/langUsage/language"	"TEI/teiHeader/profileDesc/langUsage/language"	0.79312587	1	0.2	1	0.1
6	"TEI.2/teiHeader/profileDesc/langUsage/language"	"TEI/teiHeader/profileDesc/langUsage"	0.7273346	0.7777778	0.2	0.8	0.1
7	"TEI.2/teiHeader/profileDesc/langUsage/language"	"TEI/teiHeader/profileDesc/langUsage/language"	0.70514625	0.7777778	0.2	0.8	0.1
8	"TEI.2/text/body/div1/head/placeName"	"TEI/text/body/head/placeName"	0.7046045	1	0.2	0.8333333	0.1
9	"TEI.2/text/body/div1/head"	"TEI/text/body/head"	0.69373125	1	0.2	0.8	0.1
10	"TEI.2/text/body/div1/head/date"	"TEI/text/body/head/date"	0.6687381	1	0.2	0.8333333	0.1
11	"TEI.2/teiHeader/profileDesc"	"TEI/teiHeader/profileDesc/langUsage"	0.6502725	0.090909064	0.2	0.75	0.1
12	"TEI.2/teiHeader/profileDesc/langUsage"	"TEI/teiHeader/profileDesc"	0.6502725	0.090909064	0.2	0.75	0.1
13	"TEI.2"	"TEI"	0.6486199	0.6	0.2	1	0.1
14	"TEI.2/text/body"	"TEI/text/body/div"	0.6261411	0	0.2	0.75	0.1
15	"TEI.2/text"	"TEI/text/body"	0.6248387	0	0.2	0.6666667	0.1
16	"TEI.2/text/body"	"TEI/text"	0.6248387	0	0.2	0.6666667	0.1
17	"TEI.2/text/body/div1"	"TEI/text/body/div"	0.61472905	0.75	0.2	1	0.1
18	"TEI.2"	"TEI/text"	0.60800606	0	0.2	0.5	0.1
19	"TEI.2/text"	"TEI/text/body/div"	0.6011411	0	0.2	0.5	0.1
20	"TEI.2"	"TEI/text/body"	0.5913394	0	0.2	0.33333334	0.1
21	"TEI.2"	"TEI/text/body/div"	0.57597506	0	0.2	0.25	0.1
22	"TEI.2/text/body"	"TEI/text/body/div/ab"	0.5704089	0	0.2	0.6	0.1
23	"TEI.2/teiHeader/profileDesc/langUsage/language"	"TEI/teiHeader/profileDesc"	0.56996083	0.090909064	0.2	0.6	0.1
24	"TEI.2/text"	"TEI/text/body/div/ab"	0.5504089	0	0.2	0.4	0.1
25	"TEI.2/teiHeader/profileDesc"	"TEI/teiHeader/profileDesc/langUsage/language"	0.54777247	0.090909064	0.2	0.6	0.1
26	"TEI.2/teiHeader"	"TEI/teiHeader"	0.5390482	1	0.2	1	0.1
27	"TEI.2"	"TEI/text/body/div/ab"	0.5302397	0	0.2	0.2	0.1
28	"TEI.2/teiHeader/fileDesc/sourceDesc"	"TEI/teiHeader/fileDesc/sourceDesc"	0.5	1	0.2	1	0.1

Nr.	Cosine Sim	Cosine Wght	Dice Sim	Dice Wght	Abs Freq Sim	Abs Freq Wght	Rel Freq Sim	Rel Freq Wght	Content Sim	Content Wght
1	0.7420648	0.25	0.7205688	0.25	0	0	1	0.1	0.9251362	0.1
2	0.7420648	0.25	0.7205688	0.25	0	0	1	0.1	0.9251362	0.1
3	0.6505848	0.25	0.7777778	0.25	0	0	1	0.1	1	0.1
4	0.6505848	0.25	0.7777778	0.25	0	0	1	0.1	1	0.1
5	0.6505848	0.25	0.7777778	0.25	0	0	0.36035156	0.1	1	0.1
6	0.6505848	0.25	0.7777778	0.25	0	0	0.34688348	0.1	1	0.1
7	0.6505848	0.25	0.7777778	0.25	0	0	0.125	0.1	1	0.1
8	0.8305862	0.25	0.44731182	0.25	0	0	0.017966378	0.1	1	0.1
9	0.7589173	0.25	0.49285588	0.25	0	0	0.017964102	0.1	0.98991555	0.1
10	0.61584634	0.25	0.5250375	0.25	0	0	0.017983193	0.1	0.98385537	0.1
11	0.6505848	0.25	0.7777778	0.25	0	0	1	0.1	1	0.1
12	0.6505848	0.25	0.7777778	0.25	0	0	1	0.1	1	0.1
13	0.31904963	0.25	0.6224798	0.25	0	0	1	0.1	0.9323755	0.1
14	0.74062157	0.25	0.6945291	0.25	0	0	1	0.1	0.92353415	0.1
15	0.7420648	0.25	0.7205688	0.25	0	0	1	0.1	0.9251362	0.1
16	0.7420648	0.25	0.7205688	0.25	0	0	1	0.1	0.9251362	0.1
17	0.7108302	0.25	0.36762515	0.25	0	0	0.017952314	0.1	0.9331995	0.1
18	0.7420585	0.25	0.719879	0.25	0	0	1	0.1	0.9252168	0.1
19	0.74062157	0.25	0.6945291	0.25	0	0	1	0.1	0.92353415	0.1
20	0.7420585	0.25	0.719879	0.25	0	0	1	0.1	0.9252168	0.1
21	0.7406133	0.25	0.6938409	0.25	0	0	1	0.1	0.9236148	0.1
22	0.8086585	0.25	0.5553584	0.25	0	0	0.7742807	0.1	0.919766	0.1
23	0.6505848	0.25	0.7777778	0.25	0	0	0.34688348	0.1	1	0.1
24	0.8086585	0.25	0.5553584	0.25	0	0	0.7742807	0.1	0.919766	0.1
25	0.6505848	0.25	0.7777778	0.25	0	0	0.125	0.1	1	0.1
26	0.1666018	0.25	0.00028782673	0.25	0	0	1	0.1	0.9732577	0.1
27	0.80864906	0.25	0.5546589	0.25	0	0	0.7742807	0.1	0.91984665	0.1
28	0	0.25	0	0.25	0	0	1	0.1	1	0.1

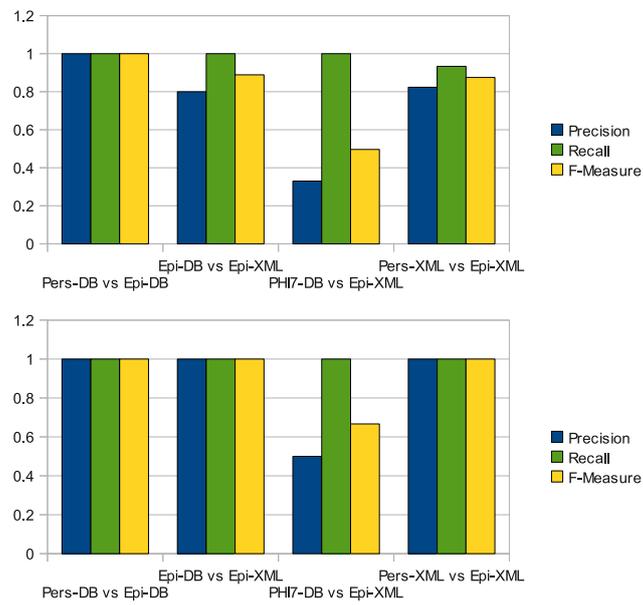
Anlage D: Beispielausgabe bis 50% Ähnlichkeit in zwei Teilen.

ABSCHNITT 8: ANLAGEN

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE TEI.2 SYSTEM "/usr/local/epiduke/data/dtd/tei-epidoc.dtd">
<TEI.2 id="o.camb.26" n="0019;26">
<teiHeader status="new" type="text">
  <fileDesc>
    <titleStmt>
      <title n="meta.73506" level="m" type="main">keiner</title>
      <title level="m" type="main">o.camb.26</title>
    </titleStmt>
    <publicationStmt></p>
    </publicationStmt>
    <sourceDesc><p n="o.camb.26">IDP generated DDbDP text</p><p n="meta.73506">IDP generated metadata from HGV database entry:
hgV73506</p><p>No corresponding HGV translation.</p>
    </sourceDesc>
    </fileDesc>
    <profileDesc><langUsage default="NO"><language id="en">English</language><language id="grc">Ancient Greek</language><language
id="la">Latin</language><language id="fr">French</language><language id="de">German</language><language id="grc-Latn">
Ancient Greek in Latin script</language><language id="cop">Optic</language><language id="it">Italienisch</language><language
id="es">Spanisch</language><languageid="el">Griechisch</language></langUsage><handList><hand id="m2"/><hand id="m3"/><hand
id="m1"/></handList><textClass><keywords><term n="meta.73506"><rs type="textType" cert="high">Name</rs><rs type="textType"
cert="high">Magie (?)</rs></term></keywords></textClass>
    </profileDesc>
    <revisionDesc><change n=""><date>2008-12-23</date><respStmt><name>IDP</name></respStmt><item>Automated split from transcoder
files</item></change><change n="meta.73506"><date>22.04.1999</date><respStmt><name>HGV</name></respStmt><item>Record created
</item></change><change n="meta.73506"><date>13.11.2007</date><respStmt><name>HGV</name></respStmt><item>Record last
modified</item></change><change n="meta.73506"><date>2009-03-20</date><respStmt><name>IDP</name></respStmt><item>Crosswalked
to EpiDoc XML</item></change>
    </revisionDesc>
  </teiHeader>
  <text>
    <body>
      <head lang="en">
        <date>323-30BC</date><placeName cert="high" full="yes">?</placeName>
      </head>
      <div type="edition" lang="grc" org="uniform" part="N" sample="complete">
        <ab part="N"><lb n="1"/>
          Ἐρμ
          <unclear reason="undefined" cert="high">
            ó
          </unclear>
          φιλός
          <lb n="1"/><gap unit="line" extent="unknown" reason="illegible"/>
        </ab>
        <div n="meta.73506" type="description" org="uniform" part="N" sample="complete"><p><rs type="material" cert="high">Ostrakon
</rs></p></div><div n="meta.73506" type="commentary" subtype="textDate" org="uniform" part="N" sample="complete"><p><date
type="textDate" notBefore="-100" notAfter="-1">I v. Chr.</date></p></div><div n="meta.73506" type="commentary" subtype="general"
org="uniform" part="N" sample="complete"><p><Descr.</p></div><div n="meta.73506" type="bibliography" subtype="principalEdition"
org="uniform" part="N" sample="complete"><listBibl default="NO"><bibl type="publication" subtype="principal" default="NO">
<title level="s" type="abbreviated">O. Camb.</title><biblScope type="numbers">26</biblScope></bibl><bibl type="Trismegistos"
default="NO"><biblScope type="numbers">73506</biblScope></bibl><bibl type="DDbDP" default="NO"><series>0019</series>;
<biblScope type="numbers">26</biblScope></bibl></listBibl><p>Perseus links: <xref type="Perseus" href="0019;26" evaluate="all"
from="ROOT" targOrder="Y" to="DITTO">Text in Perseus</xref></p></div><div n="meta.73506" type="bibliography" subtype=
"illustrations" org="uniform" part="N" sample="complete"><p>keine</p></div><div n="meta.73506" type="history" subtype=
"locations" org="uniform" part="N" sample="complete"><p>unbekannt</p>
        </div>
      </body>
    </text>
  </TEI.2>
```

Anlage E: o.camb.26.xml aus Epiduke.

ABSCHNITT 8: ANLAGEN



Anlage F: Precision, Recall und F-Measure der eindeutigen Ergebnisse, zunächst ohne Gewichtung, anschließend mit.

9 Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.

Ort

Datum

Unterschrift