Termclustering: Ähnlichkeitsmaße und Clusteringmethoden

Petra Prochazkova * Peter Kolb †

10. Dezember 2007

Zusammenfassung

Diese Arbeit liefert Erkenntnisse über den Einsatz von Ähnlichkeitsmaßen und Clusteringmethoden beim automatischen Clustern von Fachbegriffen, indem eine Auswahl von Maßen und Clusteringmethoden auf ihre Tauglichkeit für diese Aufgabe getestet wird. Ein solches automatisches Clustern liefert einen Beitrag zum effektiven Terminologieaufbau, der bislang vornehmlich manuell erfolgte.

1 Einführung

Um eine hohe Qualität von Fachtexten im Hinblick auf die verwendete Terminologie zu gewährleisten, bedarf es beim Verfassen technischer Dokumente terminologischer Datenbanken, mit deren Hilfe die verwendete Terminologie auf ihre Konsistenz überprüft werden kann. Dies dient dazu, die Texte für den Leser leichter verständlich zu machen und auch die automatische Übersetzung zu vereinfachen. Eine Terminologiedatenbank erfasst idealerweise alle Varianten eines Terms wie Synonyme und andere Schreibvarianten (*Audioeingang* vs. *Audio-Eingang*), wobei die bevorzugte Variante markiert ist. Die manuelle Ermittlung solcher Varianten ist allerdings mühsam, zeitaufwendig und fehleranfällig. Ein automatisches Clustern von Schreibvarianten wäre wesentlich effizienter und kostensparender als ein rein manuelles Vorgehen.

2 Vorgehen

Ausgehend von einer zuvor aus einem Fachkorpus extrahierten Wortliste mit 3429 Termini, werden alle Termini untereinander automatisch auf ihre Stringähnlichkeit untersucht. Diejenigen Termvarianten, deren Ähnlichkeit über einem bestimmten Schwellenwert liegt, werden im Anschluss daran mit einer geeigneten Clusteringmethode automatisch zu Äquivalenzklassen zusammengefasst.

*Institut für Linguistik, Universität Potsdam, petra.prochazkova@gmail.com
†Institut für Linguistik, Universität Potsdam, kolb@ling.uni-potsdam.de

2.1 Stringähnlichkeit

Um die Stringähnlichkeit zwischen zwei Wörtern zu berechnen, wurde im Laufe der Jahre eine ganze Reihe von Verfahren entwickelt. Zu den bekanntesten gehören die *Edit-Distanz* (*Levenshtein* 1966), buchstabenbasierte *n-Gramm*-Verfahren (s. *Angell et al.* 1983, *Rapp* 1997) oder *longest common subsequence* (LCS) (s. *Tiedemann* 1999). Das Projekt *secondstring*[1] bietet 23 in Java implementierte Maße (*AffineGap, CharJaccard, Jaro, JaroWinkler, Levenstein, MongeElkan, DirichletJS, Jaccard, JelinekMercerJS, SmithWaterman, TFIDF, TokenFelligiSunter, UnsmoothedJS, Level2Jaro, Level2JaroWinkler, Level2MongeElkan, Level2SLIM, Level2SLIMWinkler, Level2Levenstein, JaroWinklerTFIDF, SLIM, SLIMWinkler, SlimTFIDF*), die in der vorliegenden Arbeit daraufhin evaluiert werden, wie gut sie sich zur Bestimmung der Stringähnlichkeit zwischen zwei Schreibvarianten eines Terms eignen.

2.2 Clusteringmethode

Beim Clustern einer Termliste ist die Anzahl der resultierenden Cluster weder vorher bekannt noch will man ihre Anzahl vorgeben. Sie soll sich durch ein geeignetes Clusteringverfahren automatisch ergeben. Die bekanntesten Clusteringmethoden des flachen Clustering, wie z.B. *k-Nearest Neighbor, k-means-Algorithmus* oder *EM-Algorithmus* kommen daher nicht in Frage, denn bei ihnen müssen zum Clustern die Anzahl der Zielcluster bzw. der Iterationen bekannt sein. Verfahren wie *Clique, Single Link, Star* und *String* sind zum Termclustering prinzipiell tauglich; sie unterscheiden sich allerdings in der Anzahl und somit in der Güte der erzeugten Cluster, sowie in der Möglichkeit, überlappende Cluster erzeugen zu können. Es wird im Folgenden daher überlegt, welcher der vier letztgenannten Algorithmen zum Termclustering am Geeignetsten ist.

Mit der *Clique*-Methode kommen in das aktuell generierte Cluster nur die Termini, die zu *allen* Termini des Clusters ähnlich sind. Mit dieser Methode wird folglich eine hohe Precision der Ergebnisse erreicht, einhergehend mit der Generierung einer hohen Anzahl kleiner Cluster. Die Bedingung, alle Termini im Cluster müssten sich ähneln, ist zum Clustern von Termini zu restriktiv - mit der Konsequenz, dass nicht alle Schreibvarianten eines Terms den Weg in ein Cluster finden und somit das Recall zu niedrig ist. Das Herabsetzen des Ähnlichkeits-Schwellenwertes würde dem abhelfen, aber nur durch die Inkaufnahme unerwünschter Termini im Cluster, wodurch die Precison leiden würde.

Mit der *Single Link*-Methode kommen in das aktuell generierte Cluster iterativ alle Objekte, die zu mindestens einem Term im Cluster ähnlich sind. Mit dieser Methode ist folglich eine niedrige Precision der Clusterergebnisse zu erwarten, einhergehend mit der Generierung einer geringen Anzahl großer Cluster. Die Bedingung, dass ein Term im Cluster nur einem einzigen weiteren Term im Cluster ähneln muss, ist zum Clustern von Termini zu schwach, mit der Auswirkung, dass zu lange Ketten mit Termini gebildet werden, die zueinander teilweise überhaupt nicht ähnlich sind. Der Nachteil dieses Verfahrens ist auch die Unfähigkeit, überlappende Cluster zu bilden - jeder Term kann nur einem Cluster zugeordnet sein.

Mit der *String*-Methode kommen in das aktuell generierte Cluster nur die Termini, die zum zuletzt hinzugefügten Term im Cluster ähnlich sind. Die Clusterergebnisse sind bei Verwendung dieser Methode schwer abzuschätzen; es kann eine geringe bis hohe Anzahl von Clustern erzeugt werden; dies variiert stark

[1] http://secondstring.sourceforge.net

mit den zu clusternden Termini.

Mit der *Star*-Methode kommen in das aktuell generierte Cluster nur die Termini, die im Gegensatz zur String-Technik dem *ersten* Term im Cluster ähnlich sind. In der Anzahl der erzeugten Cluster liegt diese Methode zwischen Clique und Single Link und erlaubt Termini auch, in mehreren Clustern zu landen. Diese Methode eignet sich zum Clustern der Termini gut. Sie bildet weder zu kurze Ketten wie Clique noch zu lange Ketten wie Single Link. Die entstandenen Cluster hängen zwar von der Reihenfolge der Termini in der Liste ab, was nicht zum optimalsten Clusterergebnis führen muss, aber sie sind noch deutlich besser als die Cluster der anderen beschriebenen Methoden. Durch diese Clusteringmethode kann es allerdings passieren, dass ein Term, der in mehrere Cluster gehört, dennoch nicht in jedem Cluster ist, weil der Term, wenn er gerade der *Star* eines Clusters (der erste Term im Cluster) ist, dem Algorithmus nach in kein anderes Cluster mehr aufgenommen wird.

Aufgrund der genannten Vor- und Nachteile der verschiedenen Methoden wird der Star-Algorithmus zum Clustern der Termini ausgewählt. Dieses Verfahren wird in allen folgenden Evaluierungen eingesetzt. Für eine detaillierte Beschreibung des Star-Algorithmus und der übrigen angesprochenen Clusteringverfahren siehe z.B. *Kowalski* 1998, Kapitel 6.

3 Evaluation

3.1 Vorgehen

Die Evaluation der Clusterresultate erfolgt gegen eine zuvor manuell geclusterte Liste (Goldstandard) mit 3429 Termini in 2802 Clustern. Evaluiert werden die verschiedenen Stringähnlichkeitsmaße in Kombination mit dem Star-Clustering-Verfahren.
Um die automatisch gefundenen Cluster mit den manuellen Clustern des Goldstandards zu vergleichen, wird der in *Hatzivassiloglou* 1996 vorgeschlagene Ansatz benutzt. Dabei werden zu jedem Cluster alle möglichen Paare aus den enthaltenen Termini gebildet, und diese Paare dann auf Korrelation verglichen. Für alle Ähnlichkeitsmaße und Schwellenwerte (S) in 0.01-Schritten werden mit dem genannten Evaluationsmaß das jeweilige Recall (R), die Precision (P) und das F-Measure (F) berechnet. Neben einem hohen F-Measure wird ein möglichst hohes Recall angestrebt, auch wenn das auf Kosten der Precision geht, denn bei der manuellen Nachbearbeitung der Clusterergebnisse ist es einfacher, inkorrekte Termini aus den Clustern zu entfernen, als für jedes Cluster die gesamte Liste nach fehlenden Termini durchsuchen zu müssen.

3.2 Resultate

Die Maße im Projekt secondstring können danach unterteilt werden, ob es sich bei ihnen um zeichenbasierte, tokenbasierte oder eine Kombination aus beiden, d.h. hybride Maße handelt. Die Evaluation ergab, dass die zeichenbasierten und die hybriden Maße geeigneter sind zwei Termini als ähnlich einzustufen als die tokenbasierten Maße.

Die Tabelle 1 zeigt, dass die Wahl des Stringähnlichkeitsmaßes einen sehr großen Einfluss auf die Qualität des Ergebnisses ausübt. Das F-Measure schwankt für die verschiedenen Ähnlichkeitsmaße zwischen 0.08 und 0.54.
Mit der Kombination aus Starclustering und dem Jaro-Maß bei Schwellenwert

Ähnlichkeitsmaß	S	P	R	F
Jaro	0.86	0.41	0.66	0.50
Jaro	0.87	0.48	0.63	0.54
Jaro	0.85	0.35	0.68	0.46
JaroWinkler	0.90	0.28	0.67	0.39
SLIMWinkler	0.96	0.23	0.43	0.29
SLIM	0.95	0.20	0.40	0.26
Level2JaroWinkler	0.95	0.10	0.46	0.16
Level2Jaro	0.92	0.09	0.46	0.15
MongeElkan	0.82	0.08	0.71	0.14
Level2SLIMWinkler	0.96	0.06	0.46	0.10
Level2SLIM	0.95	0.05	0.44	0.08

Tabelle 1: Rangliste der String-Ähnlichkeitsmaße

0.86 werden die besten Clusterresultate erzielt, daher wird das Jaro-Maß hier genauer vorgestellt. Das Jaro-Maß (s. *Jaro* 1989) ist ein zeichenbasiertes heuristisches symmetrisches Ähnlichkeitsmaß, dessen Formel für die Berechnung der Ähnlichkeit zweier Strings s_1 und s_2 lautet:

$$Jaro(s_1, s_2) = \frac{1}{3}(\frac{common_{s1,s2}}{length\ s_1} + \frac{common_{s2,s1}}{length\ s_2} + \frac{common_{s1,s2} - transpositions_{s1,s2}}{commons_{s1,s2}})$$

$common_{s1,s2}$ steht für die Anzahl der Zeichen aus s_1, die auch in s_2 vorkommen und umgekehrt ($common_{s2,s1}$). Es werden nur die Zeichen gezählt, die nicht weiter als die Hälfte des kürzeren Strings entfernt sind; das wird als Intervall i bezeichnet, wobei auf eine ganze Zahl abgerundet wird. Wenn z.B. der kürzere String 11 Zeichen hat, dann werden nur die Zeichen gezählt, die maximal 5 Zeichen vom jeweiligen Zeichen entfernt sind. $length_{s1}$ und $length_{s2}$ stehen für die Länge der beiden Strings. $transpositions_{s1,s2}$ steht für die Anzahl identischer Buchstaben, die nicht an der gleichen Position stehen *und* deren Verbindungen sich mit anderen überkreuzen. Dieser Wert wird noch durch zwei dividiert und auf eine ganze Zahl abgerundet. Groß- und Kleinbuchstaben der beiden Strings werden bei der Berechnung des Jaro-Maßes gleich gewertet. Angenommen wir haben die Strings s_1: *MATRIX* und s_2: *MAGISTER*, dann ist die Länge von s_1: $length_{s1} = 6$, die Länge s_2: $length_{s2} = 8$ und das Intervall $i = length_{s1}/2 = 3$. Die Anzahl der gemeinsamen Buchstaben beider Strings s_1 und s_2 ist $common_{s1,s2} = 4$, weil sie die Zeichen M,A,T, und I gemeinsam haben. Das Zeichen R haben sie nicht gemeinsam, weil es außerhalb des Intervalls liegt. Es gibt eine Transposition: $transpositions_{s1,s2} = 1$, weil die zwei Buchstaben T und I in ihrer Reihenfolge innerhalb der einzelnen Strings vertauscht sind. Diese Werte eingesetzt in die Jaro-Formel, ergeben für die zwei Strings s_1 und s_2 einen Ähnlichkeitswert von 0.639:

$$Jaro(s_1, s_2) = \frac{1}{3}(\frac{4}{6} + \frac{4}{8} + \frac{4-1}{4}) = 0.639$$

4 Tuning der Clusterresultate

Die Clusterresultate aller untersuchten Maße lassen sich mit drei Parametern weiter verbessern.

Stringlänge. Nur diejenigen Termini werden als ähnlich eingestuft, die sich in ihrer Länge nicht zu sehr voneinander unterscheiden. Ist z.B. der kürzere Term 3 Zeichen lang, darf der längere nur 1 Zeichen mehr haben um noch als ähnlich zum kürzeren zu gelten. Je nach Termlänge werden verschiedene Längen-Abweichungen festgelegt.

Sonderzeichen. Deutsche Umlaute 'verrauschen' das Ergebnis der Ähnlichkeitsberechnung. So liefert z.B. das Jaro-Maß zu den zwei Termini *Haus* und *Häuser* die Ähnlichkeit 0.73. Wenn aber eindeutig erwünscht ist, diese zwei Termini als Varianten voneinander zu erkennen, bietet es sich an, den Umlaut *ä* (entsprechend die anderen Umlaute) bei der Ähnlichkeitsberechnung als ein einfaches *a* wahrzunehmen. Mit diesem Verfahren steigt der Ähnlichkeitswert auf 0.85. Das gleiche gilt für die Alternation *ß*/*ss*. Desweiteren können Sonderzeichen, die typischerweise in Schreibvarianten vorkommen (-,/,*Leerzeichen*), sowie das Fugenelement *-ungs* (*Anpassglied* vs. *Anpassungsglied*) durch ∅ ersetzt werden, um Termvarianten sicherer zu erkennen.

Symmetrie. Die secondstring-Maße lassen sich auch danach unterteilen, ob sich je nach Vergleichsrichtung Term1 vs. Term2 oder Term2 vs. Term1 verschiedene Ähnlichkeitswerte ergeben. Wenn ja, handelt es sich um sogenannte unsymmetrische Maße (*Level2Jaro*, *Level2JaroWinkler*, *Level2MongeElkan*, *Level2SLIM*, *Level2SLIMWinkler* und *TokenFelligiSunter*). Das Tunen besteht nun darin, den kleineren Wert der beiden Vergleiche als den berechneten Ähnlichkeitswert zu nehmen.

4.1 Resultate

Mit den Tuning-Parametern ergibt sich eine andere Rangliste als ohne Tuning, s. Tabelle 2. Die Maße, die die Sonderzeichen-, Längen- und Symmetrie-Parameter einsetzen, sind fettgedruckt. Aus der Tabelle 2 ist ersichtlich, dass das Jaro-Maß und das MongeElkan-Maß (beide mit Parametern) die besten Ergebnisse liefern. Mit Einsatz der Tuning-Parameter verbesserte sich das F-Measure des Jaro-Maßes von 0.50 auf 0.55, und das F-Measure des MongeElkan-Maßes von 0.14 auf 0.55. Einen detaillierten Vergleich des MongeElkan-Maßes mit und ohne Parameter zeigt die Abbildung 1. Abbildung 2 veranschaulicht die Ergebnisse der beiden besten Maße Jaro und MongeElkan jeweils mit Parametern.

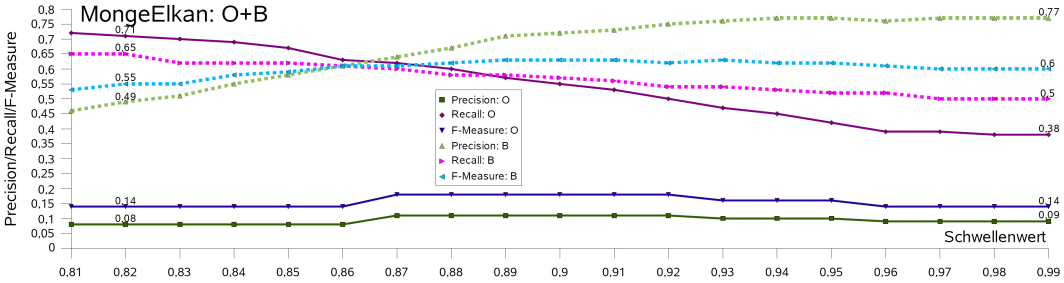

Abbildung 1: MongeElkan-Maß ohne (O) und mit Längen- und Sonderzeichenparameter (B): Precision, Recall, F-Measure

5 Schlussfolgerung

Durch theoretische Überlegungen wurde die Star-Methode als geeignetstes Verfahren zum Clustern von Termvarianten identifiziert. Die Evaluierung der String-

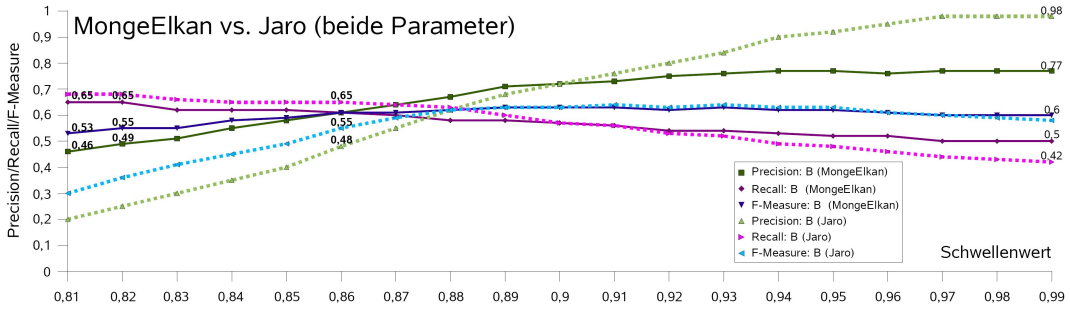

Abbildung 2: MongeElkan-Maß vs. Jaro-Maß, jeweils mit Längen- und Sonderzeichenparameter: Precision, Recall, F-Measure

Ähnlichkeitsmaß	S	P	R	F
Jaro	0.86	0.48	0.65	0.55
Jaro	0.87	0.55	0.64	0.59
MongeElkan	0.82	0.49	0.65	0.55
MongeElkan	0.81	0.46	0.65	0.53
MongeElkan	0.83	0.51	0.62	0.55
Jaro	0.86	0.41	0.66	0.50
Level2JaroWinkler	0.91	0.41	0.63	0.49
Jaro	0.87	0.48	0.63	0.54
Level2Jaro	0.86	0.39	0.64	0.48
Jaro	0.85	0.40	0.65	0.49
CharJaccard	0.91	0.43	0.51	0.46
Jaro	0.85	0.35	0.68	0.46
Level2SLIMWinkler	0.96	0.32	0.58	0.41
SLIMWinkler	0.96	0.33	0.56	0.41
SlimTFIDF	0.96	0.33	0.56	0.41
JaroWinkler	0.90	0.28	0.67	0.39
Level2SLIM	0.95	0.29	0.56	0.38
SLIM	0.95	0.30	0.55	0.38
JaroWinkler	0.90	0.39	0.64	0.48
SLIMWinkler	0.96	0.23	0.43	0.29
SLIM	0.95	0.20	0.40	0.26
Level2MongeElkan	0.83	0.14	0.56	0.22
Level2JaroWinkler	0.95	0.10	0.46	0.16
Level2Jaro	0.92	0.09	0.46	0.15
MongeElkan	0.82	0.08	0.71	0.14
Level2SLIMWinkler	0.96	0.06	0.46	0.10
Level2SLIM	0.95	0.05	0.44	0.08
JaroWinklerTFIDF	0.95	0.02	0.56	0.03

Tabelle 2: Rangliste der Maße mit Parametern (fettgedruckt) und ohne Parameter.

ähnlichkeitsmaße mit Hilfe des manuell erzeugten Goldstandards ergab, dass sich einerseits das Jaro-Maß (0.86) und andererseits das MongeElkan-Maß (0.82) mit ihren Tuning-Parametern am besten zum Clustern von Termini eignen. Das MongeElkan-Maß (*Monge, and Elkan* 1997) ist – im Gegensatz zum Jaro-Maß – erst mit den Tuning-Parametern überhaupt zum Clustern tauglich. In Übereinstimmung mit *Cohen et al.* 2003 dauert mit dem MongeElkan-Maß die Berechnung der Ähnlichkeit zwischen zwei Termini dreimal länger als mit dem Jaro-Maß.
Die Evaluation der secondstring-Maße seitens Cohen im Hinblick auf das Entfernen von Duplikaten aus Census-Daten ergab, dass das SlimTFIDF-Maß am besten für diese Aufgabe geeignet ist und die Maße Jaro, JaroWinkler und MongeElkan gut sind. Für name-matching-tasks im Allgemeinen zeigten *Porter, and Winkler* 1997, dass JaroWinkler am besten geeignet ist. Unsere Evaluation allerdings ergab, dass sich zum Clustern deutscher Termini andere Maße aus dem secondstring-Projekt besser eignen, die wir zudem durch Einsatz der vorgestellten Tuning-Parameter weiter verbessern konnten.

Ein möglicher nächster Schritt wäre, falsche Freunde in einem Cluster wie *Raum/ Traum* durch das Messen ihrer distributionellen oder konzeptuellen Ähnlichkeit zu erkennen und herauszufiltern.

Literatur

Angell, R., Freund, G., and Willett, P. (1983): "Automatic spelling correction using a trigram similarity measure". In: *Information Processing and Management*, **19**(4), 255–261.

Cohen, W., Ravikumar, P., and Fienberg, S. E. (2003): "A Comparison of String Distance Metrics for Name-matching Tasks". Technical Report, American Association for Artificial Intelligence.

Hatzivassiloglou, V. (1996): "Do we need a linguistics when we have statistics? A comparative analysis of the contributions of linguistic cues to a statistical word grouping system". In: Klavans, J., and Resnik, P., eds. , *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, pp. 67–94. Cambridge, Massachusetts: The MIT Press.

Jaro, M. A. (1989): "Advances in Record-linkage Methodology a Applied to Matching the 1985 Census of Tampa". In: *Journal of the American Statistical Association*, (84), 414–420.

Kowalski, G. (1998): *Information Retrieval Systems: Theory and Implementation*, vol. 2, *The Kluwer International Series on Information Retrieval*. Amherst: Kluwer Academic Publishers.

Levenshtein, V. (1966): "Binary Codes Capable of Correcting Deletions, Insertions and Reversals". In: *Soviet Physics-Doklady*, **10**.

Monge, A. E., and Elkan, C. (1997): "An efficient domain-independent algorithm for detecting approximately duplicate database records". In: *Research Issues on Data Mining and Knowledge Discovery*, pp. 0–.

Porter, E., and Winkler, W. (1997): "Approximate string comparison and its effect on an advanced record linkage system".

Rapp, R. (1997): "Text-detector". In: *c't-Magazin für Computertechnik*, **4**, 386.

Tiedemann, J. (1999): "Automatic Construction of Weighted String Similarity Measures". In: *Proceedings of the Joint SIGDAT Conference on Emperical Methods in Natural Language Processing and Very Large Corpora*, pp. 213–219. University of Maryland, College Park/MD.