1 Text–basiertes Informationsmanagement

Günter Neumann, LT lab, DFKI, Saarbrücken
to appear in 3rd edition of Carstensen et al. Einführung in die Computerlinguistik und Sprachtechnologie
draft Version, June 2009, daher Kommentare sehr willkommen.

1.1 Überblick

1.1.1 Allgemeines

Dank der Infrastruktur des Internets und des World Wide Webs (WWW) steht aktuell eine quasi schier unbegrenzte Menge an textuellen Quellen in Form von freien Texten und semistrukturierten Dokumenten oder Strömen solcher Quellen zur Verfügung. Dies umfasst nicht nur die mittlerweile klassischen Quellen, wie z. B. Webseiten, Emails, Nachrichtenartikel, wissenschaftliche Publikationen, Bücher, etc., sondern auch sehr dynamische Textquellen, wie Blogs, Diskussionsforen oder kollaborative Portale, wie Wikipedia und andere Wikis (siehe hierzu auch Abschnitt **??**).

Der Einsatz von Suchmaschinen im WWW (zusammen mit Annotationen und Verknüpfungen in den Dokumenten) ist aktuell die gängigste Methode, diese textuellen Quellen für den eigenen Informationsbedarf inhaltlich zu erschließen. Dabei ist die Formulierung von Informationsbedürfnissen und die inhaltliche Erschließung immer noch recht beschränkt. Im Wesentlichen geschieht dies durch Angabe von Schlüsselwörtern, deren Bestimmung in Textquellen und durch Berechnung einer möglichst optimalen Anordnung der Treffer, d. h., der gefundenen Dokumente. Gerade im Kontext des globalen, multilingualen Webs sind die wissenschaftlichen und technischen Herausforderungen auch hierbei noch enorm, vgl. hierzu Belew [2000] und Manning et al. [2008].

Allerdings liegt der Schwerpunkt der inhaltlichen Erschließung und die damit verbundene kognitive Last immer noch beim Benutzer solcher Technologien. Er muss letzlich entscheiden, ob die gelieferten Dokumente tatsächlich relevant und hilfreich sind und die gesuchte Information enthält oder zumindest Hinweise darauf. Und er muss sich letztlich selber "Gedanken machen", wie eine bessere Suchanfrage zu formulieren ist, damit eine möglichst exakte Beantwortung seiner Informationsanfrage möglich wird. Aufgrund der stetigen Zunahme an textuellen Dokumenten (insbesondere auch durch das Web 2.0) ist es aber ein zentrales Problem, dass die Informationsssuche- und sichtung einen erheblichen Zeitaufwand mit sich bringt und letztlich auch aus wirtschaftlicher Perspektive hohe Kosten erzeugt ("Time is money"). Um die Informationsflut einigermaßen in den Griff zu bekommen, erscheint ein gezielter, fokussierter Zugang auf die inhaltlichen Informationen von Texten nicht nur hilfreich, sondern notwendig.

Im Bereich der Sprachtechnologie sind hierbei in den letzten Jahren eine Reihe von Konzepten und Technologien exploriert und entwickelt worden, insbesondere in den Bereichen des Text Minings, der Informationsextraktion und

-integration, der semantischen Suche und Fragebeantwortung und der Informationspräsentation. Aktuell ist ein starke Konvergenz zwischen diesen Bereichen zu beobachten, insofern als viele Teillösungen gemeinsamen Ursprung haben, wie z. B. die linguistische Merkmalsextraktion oder die Identifikation von relevanten Entitäten und Relationen zwischen ihnen.

Diese Konvergenz unter der Bezeichnung **"Text–basiertes Informationsmanagement" (TIM)** aufzuzeigen und zu beschreiben ist Gegenstand dieses Unterkapitels.

1.1.2 Information

Um besser verstehen zu können, was "Text–basiertes Informationsmanagement" meint, wollen wir kurz klären, wie wir im Kontext dieses Artikels den Begriff "Information" verstehen. Man muss hierbei vorausschicken, dass dies in der Tat keine einfache Sache ist, da es noch keine eindeutige und zufriedenstellende Definition des Begriffes gibt, vgl. Meadow [1992] und Floridi [2005]. Wir werden daher den Begriff hier nur soweit klären, wie er für das weitere Verständnis wichtig ist.

In den letzten Jahrzehnten wird meist eine Definition von **Information** als "Daten + Bedeutung" bevorzugt, vgl. Floridi [2005]. Vereinfacht betrachten wir "Information" als eine Menge von diskreten, wohlstrukturierten und bedeutungsvollen Daten. Daten sind quasi die physikalischen Implementationen von Symbolen (z. B. in einer Datenbank oder als Druckertinte auf Papier); **wohlgeformt** meint, dass die Daten in einer Struktur organisiert sind; **bedeutungsvoll** meint, dass diese organisierte Anordnung von Daten von einem Benutzer gelesen und verstanden werden kann. Unter **Lesen** wollen wir verstehen, dass die Datenquelle von einem Benutzer mit Hilfe seiner Sensoren empfangen, internalisiert und strukturiert werden kann, sodass auf deren Basis dann der eingebettete Inhalt bestimmt werden kann.

An einem einfachen Beispiel wollen wir das kurz klären. Betrachten wir das Symbol "28081749". Ohne weitere "Information" können wir dies als eine Zeichenfolge bestehend aus natürlichen Zahlen "verstehen". (Aber aufgemerkt: Damit wir dies tun können, müssen wir natürlich "wissen", was "natürliche Zahlen" und auch "Zeichenfolgen" sind.) Allerdings kann man diese Zeichenfolge auch interpretieren als "Goethes Geburtsdatum" (Goethe wurde am 28. August 1749 geboren) oder auch als Breitengrad südlich von "New Delhi". Das "28081749" Goethes Geburtsdatum meint, mag man auch dadurch ableiten, dass das Symbol etwa in einem Satz wie "Goethe wurde am 28081749 geboren." vorkommt oder direkt als Element in einer Datenbank gepeichert ist, z. B. als Attribut–Wert–Paar $\langle$Geburtsdatum = 28081749$\rangle$, das wiederum Teil eines komplexeren Eintrags über Wolfgang von Goethe sein kann.

Dieses Beispiel macht deutlich, dass Daten mehrere Interpretationen haben können und nur eindeutig interpretierbar sind, wenn Wissen und Kontext herangezogen wird, vgl. auch Abschnitt **??**. Da Texte bekanntermaßen hochgradig mehrdeutig sind, ist das Informationspotential in Texten entsprechend hoch. Wir wollen daher als **relevante Information** den Anteil an der potenti-

ellen Information bezeichnen, der von einem Benutzer als "nützlich" betrachtet wird. Man beachte, dass die Bewertung, ob Information relevant ist oder nicht, benutzer– und situationsabhängig ist, wogegen "Information" dies per se nicht ist, d. h., Information kann auch irrelevant sein.

Wichtig ist aber zu beachten, dass die Information stets eingebettet in einer physikalischen Implementation auftritt, sei es als Datenbank, als Bild, als Video oder eben als Text.

1.1.3 Text–basiertes Informationsmanagement—ein Walkthrough

Wesentliche Hauptaufgaben eines Informationsmanagementsystems sind:

- Die Information, die in digitalisierter Form als Daten repräsentiert ist, zu verwalten.
- Die zur Beantwortung einer Anfrage notwendige relevante Information zu bestimmen und einem Benutzer in verständlicher Form zur Verfügung zu stellen.

Im Falle eines TIM-Systems ist Information durch natürliche Sprache in Texten kodiert und muss entsprechend in Daten überführt werden. Je nachdem, wie tief und umfassend diese Strukturierung vorgenommen wird, muss auch eine Benutzeranfrage entsprechend interpretiert und strukturell analysiert werden.

Wir wollen an einem komplexeren Beispiel die wichtigsten Eigenschaften eines TIMs zeigen, die dann im späteren Verlauf des Artikels detaillierter beschrieben werden. Folgende Situation sei für das Beispiel angenommen: Ein Benutzer des TIMs — nennen wir sie im weiteren USER — möchte Informationen zu Umsatzmeldungen von Unternehmen aus der IT Branche sammeln, vielleicht, weil sie selber ein Unternehmen gründen möchte, oder weil sie diese Informationen nutzen möchte, um versteckte Beziehungen über mögliche Verflechtungen von Unternehmen zu bestimmen. Nehmen wir zusätzlich an, USER verfüge bereits über eine Datenbank mit Instanzen solcher Informationen (und gegebenenfalls Verweise auf die textuellen Quellen). Folgendes Beispiel zeigt einen Eintrag aus dieser Datenbanktabelle namens UMSATZMELDUNGEN:

Unternehmen	Jahr	Größe	Betrag	Tendenz	Differenz
Compaq	2001	Umsatz	8.4 Mrd. USD	-	16.67%

Hierbei repräsentiert das Attribut *Unternehmen* den Namen eines Unternehmens, *Jahr* das Jahr der Meldung und *Betrag* den mitgeteilten Umsatzwert; das Attribut *Größe* kodiert, ob es sich um einen Umsatz, Gewinn oder Verlust handelt und *Tendenz* entweder eine Zunahme (+) oder Abnahme (-) beschreibt. *Differenz* kodiert den prozentualen Unterschied zum Vorjahr.

Ziel ist es nun, durch Analyse von Webdokumenten automatisch diese Tabelle zu erweitern oder zu aktualisieren. Welche Schritte sind hierzu im Einzelnen notwendig? Die wohl wichtigsten sind:

1. Analyse der Informationsanfrage
2. Bestimmung von relevanten Dokumenten
3. Bestimmung der relevanten Textpassagen
4. Extraktion von Attributwerten und –beziehungen
5. Erzeugen neuer Tabelleneinträge
6. Integration in bestehende Tabellen
7. Präsentation der Ergebnisse

Wir werden nun die einzelnen Schritte an Hand unseres obigen Beispielszenarios durchspielen, wobei wir annehmen, dass es bereits ein System gibt (mit Namen TIMPLE – *Text–based Information Management Example*), dass die komplette Aufgabe automatisch löst, sobald USER eine Informationsanfrage an das System sendet. Nehmen wir an, USER formuliere ihre Anfrage direkt als natürlichsprachliche Frage "Welche Firmen aus der Computerbranche steigerten ihren Umsatz?". Als ersten Schritt analysiert TIMPLE die Anfrage derart, dass eine Suche nach relevanten Dokumenten gestartet werden kann. Nehmen wir an, dass zur Dokumentenbestimmung TIMPLE eine oder mehrere Suchmaschinen einsetzt, etwa Ask (http://de.ask.com/), Google (http://www.google.de), MSN LiveSearch (http://www.live.com/) oder Yahoo (http://de.yahoo.com/). Dann ist die einfachste Art der Analyse, die Frage direkt an eine Suchmaschine weiterzuleiten. Die Antwort ist dann in Form einer sortierten Liste von Treffern, etwa wie folgt (Es handelt sich hierbei tatsächlich um das Ergebnis der obigen Anfrage, die der Autor am 20.11.2008, ca. 15:00 Uhr durchgeführt hat. Es werden die ersten drei von 153 Treffern, in der Darstellung entsprechend angepasst, gelistet.):

1. Dell hängt Rest der **Computerbranche** ab - Unternehmen - IT ...
   Die US-**Firma steigerte** im zweiten Quartal zwar den **Umsatz** um 8,1 % auf 1,55 ... hängt stark von der Entwicklung der Weltwirtschaft und **ihrem** Einfluss auf ...
   `www.handelsblatt.com/unternehmen/it-medien/dell-haengt-rest-der-computerbranche-ab;646067`

2. PC-Hersteller bleiben auf Wachstumskurs - Business — ZDNet.de News
   Die Texaner **steigerten ihren** Gewinn im Vergleich zum Vorjahr um ein Viertel auf 846 Millionen Dollar (656 Millionen Euro). Der **Umsatz** kletterte um 18 ...
   `www.zdnet.de/news/business/0,39023142,39127741,00.htm`

3. TAM - Energienews täglich aktuell - Energienachrichten per ...
   In den ersten neun Monaten 2008 **steigerte** die schwedische Vattenfall wegen gestiegener Preise **ihren Umsatz** um knapp 12% ...
   `www.tam.de/index.asp?ACTION=1`

Jeder Treffer besteht aus einer Textüberschrift, einem Textausschnitt und dem Link zur Webquelle. Wie bestimmt eine Suchmaschine die Dokumente? Nun, unter anderem auch dadurch, dass es einen wortbasierten Index für alle verfügbaren Webdokumente verwaltet. Damit ist es möglich, Dokumente dadurch auszuwählen (zu "erschließen", engl. **document retrieval**), indem man im Index unter den Suchwörtern nachschlägt. Im Textausschnitt (auch: *Snippet*), werden die Wörter, die mit denen aus der Anfrage übereinstimmen, hervorgehoben. Man beachte, dass nicht alle Wörter aus der Frage auch in den Dokumenten vorkommen müssen. So kommt das Wort "Computerbranche" nicht in den Treffern 2 und 3 vor. Man sieht aber auch, dass morphologische Varianten eines Wortes (z. B. "ihren" in der Anfrage und "ihrem" in Treffer 1) scheinbar keine Probleme bereiten.

Obwohl für dieses Beispiel nur eine kleine Dokumentenmenge bestimmt wurde (153 Dokumente), sind offensichtlich nicht alle Dokumente gleichermaßen relevant. So thematisiert Treffer 3 Umsatzmeldungen von Unternehmen aus dem Energiesektor. Daher **klassifiziert** TIMPLE die Dokumente (nachdem es diese mittels des Weblinks heruntergeladen und in eine uniforme interne Dokumentenrepräsentation überführt hat) in solche, die zur Klasse "Computerbranche" gehören und solche die zu anderen Klassen gehören. Es ist offensichtlich, dass dies zwar ein semantischer Prozess ist, in der Regel aber auf einer sehr großen Textmenge durchzuführen ist und als Vorauswahl für die nachfolgenden Informationsextraktionprozesse dient. Daher muss diese Klassifikation sehr robust, schnell und dennoch möglichst genau durchgeführt werden. TIMPLE ist zum Glück dazu in der Lage, da es auf der Basis von früheren Analysen automatisch gelernt hat, welche Wörter und wortbasierten Merkmale helfen, die Klassenzuordnung durchzuführen.

Nehmen wir an, dass auf diese Weise klar ist, welche Dokumente "Umsatzmeldungen" enthalten und welche nicht. In der Regel wird ein solches Dokument aber nicht nur über Textpassagen verfügen, in denen die Informationen über Umsatzmeldungen gebündelt vorkommen, sondern der Text wird auch andere Informationen kodieren, die aus Sicht der Umsatzmeldung irrelevant sind. Daher ist es ein sinnvoller nächster Schritt für TIMPLE die Textpassagen zu identifizieren, die den Umsatzmeldungen "unmittelbar" entsprechen. Im Prinzip bedeutet dies, dass ein Text in eine Folge von Texteinheiten zu überführen ist, sodass für jede Texteinheit beurteilt werden kann, ob sie relevant oder nicht relevant zur Beantwortung der Informationsanfrage ist. Oder anders formuliert, es sollen genau die **Textpassagen bestimmt werden** (engl. **passage retrieval**), in denen potentielle Werte für Attribute von Umsatzmeldungen zu finden sind. Betrachten wir als Beispiel hierfür den Inhalt des Textes zum Treffer Nummer 1 (aus Platzgründen werden nur die ersten 3 der insgesamt 8 Paragrafen gezeigt, aus denen das Originaldokument besteht):

> http://www.handelsblatt.com/, 18.07.2003
>
> Größter Gewinner ist einmal mehr der Computerbauer Dell. Die Amerikaner sind weltweit die Nummer eins und wesentlich stärker gewachsen als der nächste Verfolger, Hewlett-Packard. "Die Kunden

> honorieren unser effizientes Geschäftsmodell, das auf Händler ganz verzichtet", sagte der Deutschland- Chef von Dell, Mathias Schädel, dem Handelsblatt. Das US-Unternehmen vertreibt seine Produkte ausschließlich über eigene Mitarbeiter, zumeist per Internet.
>
> Doch nicht nur Dell kann zufrieden sein. *Eine Mixtur aus wachsendem Dienstleistungsgeschäft, Kostensenkungen und erfolgreichen Akquisitionen brachte Wettbewerber IBM im zweiten Quartal deutlich verbesserte Ergebnisse. Zwischen April und Juni stiegen der Umsatz um 10% auf 21,6 Mrd.$ und der Reingewinn auf 1,7 Mrd.$. Sonderlasten in Höhe von 1,4 Mrd.$ hatten den Vorjahresgewinn auf 56 Mill.$ gedrückt.*
>
> *Der kultige Nischenanbieter Apple dagegen steht nicht ganz so gut da. Die US-Firma steigerte im zweiten Quartal zwar den Umsatz um 8,1% auf 1,55 Mrd.$. Der Nettogewinn ging jedoch von 32 auf 19 Mill.$ zurück.* Apples Ergebnisse übertrafen damit die pessimistischen Erwartungen der Analysten. Apple gilt seit langem als innovativ und experimentierfreudig, der weltweite Marktanteil lag 2002 aber nur noch bei 2,3% gegenüber 8,3% vor zehn Jahren.

Die kursiv hervorgehobenen Stellen seien die relevanten Textpassagen. Wie könnte TIMPLE diese Aufgabe leisten? Zum Einen scheint klar, dass TIMPLE Texte in Einheiten, wie Zeichenfolge, Sätze, Paragrafen etc. strukturieren können muss. Zum Anderen scheint aber auch klar, dass TIMPLE "wissen muss", wie die Struktur der Datenbanktabelle UMSATZMELDUNGEN definiert ist, d. h., es muss den Typ der Attribute kennen und die Stelligkeit der Tabelle. Die Tabelle kann auch als eine n–äre Relation betrachtet werden, wobei die einzelnen Spalten den Argumenten entsprechen. Die konkrete Relation unserer Tabelle entspricht demnach der 6–stelligen Relation

> UMSATZMELDUNGEN(UNTERNEHMEN, JAHR, GRÖSSE, BETRAG, TENDENZ, DIFFERENZ)

Das bedeutet, dass TIMPLE nicht nur diese Relation "wissen" muss, sondern auch "operationalisieren" können muss. Was heißt das? Nun, TIMPLE muss bereits bei der Bestimmung von relevanten Textpassagen zumindest teilweise Wissen über diese Relation einsetzen, damit überhaupt relevante Textausschnitte bestimmt werden können. Beispielsweise sollte in einer relevanten Textpassage zumindest der Name eines Unternehmens (z. B. "Apple") explizit oder zumindest implizit ("die US–Firma") genannt werden.

Daher betrachtet TIMPLE eine **Relation** als eine Komposition von diskreten Elementen, die es versucht, von den einfachen Elementen bis zu deren komplexer Vernetzung zu identifizieren und zu organisieren. Dies umfasst zuerst die Bestimmung aller potentiellen Werte für die verschiedenen **Attribute** (Argumente) der Relation. Unserem Beispiel folgend müssen zum Beispiel alle möglichen Firmennamen oder Geldbeträge im gesamten Text bestimmt werden, bevor in einem nächsten Schritt entschieden werden kann, welche der Kandidaten

tatsächlich Teil welcher Relation sind. Im Beispieltext werden die Computerfirmen "Dell", "Hewlett–Packkard", "IBM" und "Apple" genannt. Erst später wird sich zeigen, dass aber nur "IBM" und "Apple" von Relevanz sind, da "Dell" und "Hewlett–Packard" nicht als Wert in einer relevanten Relation genannt sind. Die Erkennung solcher spezifischen **Eigennamen** ist nicht trivial, d. h., nicht einfach durch einen direkten Zugriff auf eine eventuell vorhandene Liste aller Firmennamen zu realisieren. Zum Einen kann es sein, dass im Text verschiedene Schreibvarianten eines Firmennamens verwendet werden (z. B. "Apple", "Apple Inc.", "Apple Deutschland" etc.). Zum Anderen könnten Firmennamen auch implizit genannt werden, z.B "Apple" implizit durch die definite Nominalphrase "der US–Konzern". Das bedeutet, dass zur Erkennung von Eigennamen nicht nur alle Benennungen von Firmennamen zu identifizieren sind, sondern diese auch korrekt den denotierten Entitäten zuzuordnen sind.

Es erscheint klar, dass zumindest für die **Bestimmung von Koreferenzen** zwischen Eigennamen und Nominalphrasen eine zumindest partielle syntaktische Analyse notwendig ist, die über die Behandlung von Phänomenen auf der Wortebene hinausgeht; z. B. müssen nicht nur die Nomengruppen erkannt werden, sondern auch ihre interne Struktur, wie etwa Spezifikator-Kopf-Kongruenz. Dies wird insbesondere dann deutlich, wenn die potentiellen Attributwerte zu komplexeren Einheiten als Relation verknüpft werden müssen. Eine mögliche Strategie hier ist anzunehmen, dass alle Attributwerte, die zu einer Relation gehören, in **struktureller Nähe** zueinander auftreten. Daher wollen wir annehmen, dass TIMPLE zur weiteren Bestimmung von Relationen eine syntaktische Analyse aller relevanten Sätze durchführt, also aller Sätze, in denen mindestens ein Attributwert vorkommt. Darauf aufbauend können dann mit Hilfe von relationenspezifischen Mustern (etwa der Art "NP(FN) VP(STEIGEN(UMSATZ)) PP(UM, NP(B))", wobei z. B. NP(FN) zu Lesen ist als "Nominalphrase deren Kopfelement vom Typ Firmenname ist") die Argumentwerte bestimmt und zugeordnet werden, die zu einer Relation gehören.

Allerdings kann im Allgemeinen nicht angenommen werden, dass stets immer alle Werte einer Relation in einem Satz auftreten. Daher muss TIMPLE auch in der Lage sein, **Teilrelationen** zu identifizieren, wie z. B. für den Textabschnitt:

> Eine Mixtur aus wachsendem Dienstleistungsgeschäft, Kostensenkungen und erfolgreichen Akquisitionen brachte Wettbewerber **IBM** im zweiten Quartal deutlich verbesserte Ergebnisse. Zwischen April und Juni **stiegen der Umsatz** um **10%** auf **21,6 Mrd.$** und der Reingewinn auf 1,7 Mrd.$.

Die entsprechenden **partiellen Instanzen** sehen wie folgt aus:

Unternehmen	Jahr	Größe	Betrag	Tendenz	Differenz
IBM					
		Umsatz	21,6 Mrd. $	+	10 %

Nachdem TIMPLE also alle potentiellen möglicherweise partiell instanziierten Relationen berechnet hat, muss in einem nächsten Schritt entschieden werden, welche Teilrelationen zusammen gehören und wie die identifizierten Teile **fusioniert** werden sollen. Für das obige Beispiel wählt TIMPLE eine einfache, aber bewährte Heuristik: Es werden die Attributwerte von aufeinander folgenden Sätzen vereinigt, solange keine Inkonsistenzen entstehen. Für das obige Beispiel bedeutet dies, dass die zwei partiellen Einträge einfach verschmolzen werden können zu:

Unternehmen	Jahr	Größe	Betrag	Tendenz	Differenz
IBM	2003	Umsatz	21,6 Mrd. $	+	10 %

Man beachte, dass TIMPLE hierbei auch den Wert für das Jahr eingefügt hat, obwohl dieser nicht explizit im Text genannt wird, wohl aber in der Kopfzeile des Artikels. Es ist klar, dass dies nur eine einfache (allerdings erprobte) Heuristik ist, da im Allgemeinen hier komplexere Inferenzprozesse nötig sind, um den richtigen zeitlichen Kontext einer Relation zu bestimmen. Da TIMPLE aber stets unter großem Druck steht, die relevanten Informationen möglichst zeitnah zu liefern, muss TIMPLE oft einen Kompromiss zwischen Aktualität und Genauigkeit finden.

Bevor diese neuen Informationen aber an USER weitergeleitet werden, integriert TIMPLE die gefundenen Instanzen als neue Einträge in USERs bestehende Datenbank UMSATZMELDUNGEN. Auch dies ist kein trivialer Prozess, da die Konsistenz der gesamten Datenbank nicht beschädigt werden darf, u.a., durch doppelte Einträge, durch sich widersprechende Einträge oder durch partielle Einträge.

Nehmen wir an, dass TIMPLE auch diesen Schritt meistert, dann wäre es soweit, USER die neuen Informationen mitzuteilen. Eine einfache Meldung könnte sein: "Liebe USER, es gibt 50 neue Einträge in der Datenbank UMSATZMELDUNGEN." Solch eine Meldung kann nützlich sein, ist in der Regel aber suboptimal. TIMPLE beschließt daher, automatisch eine verständliche Zusammenfassung der neuen Einträge als Ergebnis zu liefern, mit direkter Verbindung zu den Einträgen in der Datenbank, etwa durch folgende **Textzusammenfassung**:

> Hallo USER! Ich habe 50 neue Einträge in die Datenbank UMSATZMELDUNGEN eingefügt. Für folgende Firmen liegt eine Umsatzsteigerung im Vergleich zum Vorjahr 2002 vor: *Dell*, *IBM*, ... Folgende Firmen verzeichnen dagegen einen Verlust: *Sugar&Brothers*, *SaltInc.*, , ...

Dies ist natürlich nur eine mögliche Textzusammenfassung. Eine andere Möglichkeit wäre es, direkt die relevanten Textausschnitte heranzuziehen und kohärent zusammenzufügen, eventuell durch Verwendung von Paraphrasen. Auch sind wir im gesamten Beispiel davon ausgegangen, dass die relevanten Dokumente im gesamten Web gesucht werden. Hier wäre es auch möglich, direkt (zumindest für die konkrete Anfrage) Geschäftsberichte von Firmen zu analysieren oder

spezielle Nachrichtenmagazine.

Desweiteren gingen wir davon aus, dass die Informationsanfrage in Form einer vollständigen **natürlichsprachlichen Frage** formuliert ist und dass TIMPLE dies schon entsprechend interpretieren kann. Hierzu muss TIMPLE jedoch "verstehen", dass mit der Frage das Auffüllen der Tabelle UMSATZMELDUNGEN "gemeint" ist. Dies könnte TIMPLE tatsächlich auf verschiedene Weise leisten. Eine Möglichkeit ist es, zum Einen den **Fragetyp** (es soll eine Liste von Elementen eines bestimmten Typs bestimmt werden) und zum Anderen den **erwarteten Typ der Antwort** (hier Relation UMSATZMELDUNG) zu bestimmen. Letzteres könnte einfach dadurch realisiert werden, dass das Tabellenschema direkt mit bestimmten Schlüsselwörtern assoziiert ist. Wenn die Schlüsselwörter (oder Varianten davon) in der Frage auftreten, wird das entsprechende Schema aktiviert.

In der Regel wird TIMPLE aber über viele verschiedene solcher Tabellenschemata verfügen, so dass es sinnvoll erscheint, diese systematisch in einem strukturierten Vokabular (**Ontologie**, vgl. auch **??**) zu verwalten. Daher könnte die Interaktion mit einem Benutzer auch so gestaltet werden, dass TIMPLE diese Ontologie dem Benutzer visualisiert, so dass der Benutzer einen relevanten Teilausschnitt eigenständig bestimmen kann und dann in diesem Kontext der Ontologie seine aktuelle Informationsanfrage formuliert, z. B. indem der Benutzer für einige Attribute Werte vorgibt (z. B. bestimmte Firmennamen oder Quartalsangaben etc.).

Eine weitere Möglichkeit mit TIMPLE zu interagieren wäre es, dem System eine Menge von Texten zu geben, die bereits mit Beispielen der gewünschten Informationen manuell annotiert sind, in unserem Beispiel also mit Relationen zu Umsatzmeldungen. TIMPLE kann dann dieses **Korpus** heranziehen, um mit Hilfe von **maschinellen Lernverfahren** automatisch zu erlernen, neue Texte als Umsatzmeldungen zu erkennen und die entsprechenden Attributwerte und Relationen zu finden und zu extrahieren.

All diesen verschiedenen Möglichkeiten ist gemein, dass sie die gleichen Basisaufgaben zu lösen haben, u. a. Volltextsuche, Klassifikation von Texten, Bestimmen von relevanten Textausschnitten, linguistische Merkmalsextraktion, Extraktion von Eigennamen und Relationen, sowie deren Fusion und Integration in bestehende Datenbestände. In den folgenden Abschnitten wird auf die verschiedenen Basistechnologien genauer eingegangen und es werden aktuelle Methoden und Verfahren kurz vorgestellt.

1.1.4 TIM-Grobstruktur

Die Grafik in Bild 1 zeigt die grobe Struktur eines (idealen) generischen TIM. Ein TIM vereinigt in sich die verschiedenen Perspektiven des Information–Retrieval (IR), der Informationsextraktion (IE) und der Fragebeantwortung (QA – Question Answering).

Die Hauptaufgaben im Bereich IR liegen in der Indizierung und Suche im gesamten Dokumentenbestand (**Volltextsuche**), in der Textklassifikation und im Textclustering (**Textgruppierung**). Der Schwerpunkt im Bereich QA liegt

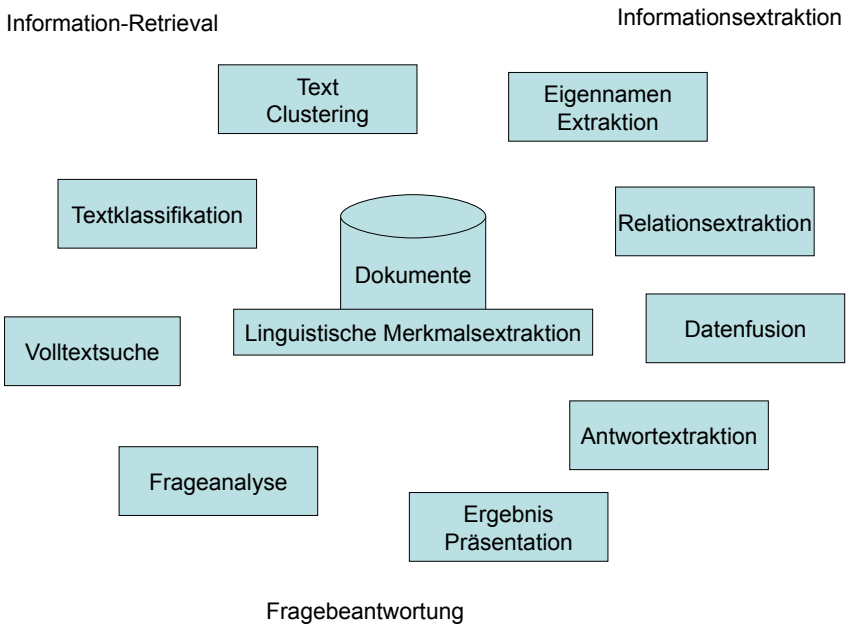

Abbildung 1: Text–basiertes Informationsmanagement: Technologieperspektive. Genaue Erläuterungen finden sich im Text.

in einer genauen Analyse der Benutzeranfrage (**Frageanalyse**) und in der Extraktion von exakten Antworten (**Antwortextraktion**) aus relevanten Textausschnitten. Im Bereich der IE liegt der Forschungsschwerpunkt in der Extraktion von **Eigennamen** und **Relationen** und in der Gruppierung und Fusion gefundener Entitäten.

Für alle drei Bereiche zentral ist natürlich eine textuelle Dokumentenbasis und vor allem die Extraktion von linguistischen Merkmalen.[1] Allerdings unterscheiden sich die Bereiche in der Größenordnung der Dokumentenmenge, die sie unmittelbar zu verarbeiten haben und damit auch in der Tiefe und Genauigkeit der linguistischen Analyse, die noch machbar ist. Für das IR gilt, dass hier sehr viel größere Datenmengen zu verwalten sind (z. B. Milliarden von Webseiten), wogegen die Dokumentenmenge für die IE und QA meist überschaubar ist (z. B. einige tausend Dokumente). Daher basieren die in der IR eingesetzten Komponenten in der Regel auf wortbasierten Merkmalen. In der QA und insbesondere in der IE ist jedoch eine umfangreichere linguistische Merkmalsextraktion not-

[1] Vergleiche hierzu Kapitel **??**, insbesondere die Abschn. **??**, **??**, **??** und **??** und aus Kapitel **??**, Abschn. **??**.

wendig, wie das obige Beispiel verdeutlicht.

Auf der anderen Seite bauen die verschiedenen Technologiezweige in immer stärkerem Maße aufeinander auf. So verwenden viele QA–Systeme direkt IR–Systeme zur Vorauswahl einer relevanten Textmenge. Dabei setzen sie auch unmittelbar Technologien aus dem Bereich der IE ein, wie z. B. die Extraktion von Eigennamen und Relationen. IE–Systeme setzen verstärkt Technologien aus dem Bereich des IR ein, insbesondere zur Bestimmung relevanter Textausschnitte und zur Gruppierungen von Daten.

1.1.5 Evaluationskriterien für TIM

Um die Güte der einzelnen Komponenten zu messen, werden oft zwei Maße, die **Präzision** und die **Vollständigkeit**, verwendet, die im Folgenden beschrieben werden:

- Die Präzision (engl. **Precision** (P)) bezeichnet den Anteil der korrekt gewonnenen Wissenseinheiten (WE) (Sätze oder einzelne Attribut-Wert Paare) im Vergleich zu den insgesamt gefundenen WE. Eine hohe Präzision bedeutet daher, dass fast alle gefundenen WE relevant sind, während eine geringe Präzision besagt, dass auch viele nicht relevante WE fälschlicherweise als relevant beurteilt wurden.

- Die Vollständigkeit (engl. **Recall** (R)) bezeichnet den Anteil der korrekt gewonnenen WE im Vergleich zu den insgesamt gewinnbaren WE. Bei einer geringen Vollständigkeit wurden viele relevante WE übersehen, während bei einer hohen Vollständigkeit fast alle relevanten WE extrahiert wurden.

Es ist schwierig, beide Parameter gleichzeitig zu optimieren: Wird eine Suche auf eine hohe Präzision hin optimiert, so steigt die Wahrscheinlichkeit, dass möglicherweise relevante Wissenseinheiten nicht erkannt werden. Optimiert man andererseits die Vollständigkeit, so steigt die Gefahr, dass Wissenseinheiten mit in das Ergebnis aufgenommen werden, die irrelevant sind. Um ein zusammenfassendes Maß für die Güte einer Methode zu schaffen, wurde das **F-Maß** definiert (in der Regel wird in folgender Gleichung β=1 gesetzt):

$$F = \frac{(\beta^2 + 1) * P * R}{\beta^2 P + R}$$

Die Güte des Ergebnisses hängt stark von der Schwierigkeit der Aufgabe ab. So berichten beispielsweise Grishman and Sundheim [1996] im Rahmen der MUC-7, dass bei einer einfachen Aufgabe, wie der Erkennung von Namen, die meisten Systeme sowohl bei der Vollständigkeit als auch bei der Präzision Werte von über 90% erreichten (das beste System erreichte R=96% und P=97%). Allerdings war bei dieser Aufgabe die Domäne der Texte stark eingeschränkt, da es sich ausschließlich um Texte aus dem Wall Street Journal handelte.

Appelt and Israel [1997] weisen darauf hin, dass empirisch festgestellt wurde, dass bei komplexen Aufgaben im Bereich der Informationsextraktion die Güte

des Ergebnisses in der Regel nicht besser als 60% bezüglich des F-Maßes ist. Eine solche Güte ist jedoch nicht so schlecht, wie es auf den ersten Blick erscheint, denn auch Menschen sind nicht in der Lage, bei der Analyse von komplexen Texten sowohl 100% Vollständigkeit als auch 100% Präzision zu erreichen. Ein Mensch erreicht bei der Erkennung von Eigennamen ein F-Maß von etwa 97%. Darüber hinaus arbeiten IE-Systeme im Gegensatz zu Menschen ermüdungsfrei, während die menschliche Leistung infolge Ermüdungserscheinungen nachlässt.

Die Maße Precision und Recall können auch aus der folgenden **Kontingenztabelle** abgeleitet werden, die im Bereich des Information–Retrievals häufig herangezogen wird:

	D ist relevant	D ist nicht relevant
D ist gefunden	a	b
D ist nicht gefunden	c	d

Hierbei meint "D ist relevant" die Anzahl der Dokumente (oder Eigennamen oder Relationen), die relevant sind, und "D ist gefunden" die Anzahl der Dokumente, die ein System als Ergebnis geliefert hat (analoges für "nicht"). Dann ist zum Beispiel a die Anzahl der Dokumente die relevant sind und vom System geliefert wurden und c die Anzahl der relevanten Dokumente, die vom System nicht gefunden werden konnte. Die Summe von a, b, c und d ist die Gesamtanzahl aller Dokumente, die in einem Test ausgewertet werden. Damit entspricht Precision P der Anzahl P=a/(a+b) und Recall R entspricht R=a/(a+c). Desweiteren lässt sich die Akkurartheit A eines Systems berechnen durch A=(a+d)/(a+b+c+d). Dies setzt natürlich voraus, dass man von jedem Dokument in einem Korpus weiß, ob es relevant ist oder nicht.

Ein weiteres häufig benutztes Maß speziell im Bereich der Fragebeantwortung ist der **Mean Reciprocal Rank** (MRR). Der **MRR** weist dort jeder Frage einen Wert zu, der gleich dem Kehrwert des Ranges der ersten korrekten Antwort der N besten Kandidaten ist, vgl. auch Abschnitt 1.4.1.

Im Gegensatz zur Akkuratheit betrachtet der MRR also mehrere Antwortkandidaten des Systems, wobei berücksichtigt wird, ob die richtige Antwort an erster, zweiter oder anderer Stelle steht. Wenn N=1 gewählt wird, entspricht das Ergebnis demnach der Akkuratheit des Systems. Der MRR ist zwar ein weicheres, allerdings aus Anwendungsperspektive praktikableres Maß als die Akkuratheit, z. B. in Fällen, wo ein Benutzer selbst die passende Antwort aus einer Menge von Kandidaten wählen kann.

1.2 Information Retrieval

Aus Sicht eines TIMs ist die wichtigste Aufgabe des IR, es einem Benutzer zu ermöglichen, *in sehr großen Textmengen sehr schnell zu suchen* (auch als **Volltextsuche** bekannt), um so rasch eine relevante Textmenge zu bestimmen. Man kann sich die Informationsverarbeitung eines TIMs auch als ein **interaktives Textzooming** vorstellen, wo ausgehend von einer initialen Suchanfrage Schritt für Schritt immer "tiefer" in die Inhalte der Texte hineingezoomt wird,

bis die "richtigen" Antworten gefunden sind. Diese Interaktion kann direkt zwischen Mensch und Maschine stattfinden, indem die Maschine in jedem Schritt eine immer feinere Auswahl von Texten bestimmt, auf deren Basis der Mensch eventuell weitere Anfragen stellt. Man kann sich diese Interaktion aber auch direkt zwischen den Komponenten eines TIMs vorstellen. Zum Beispiel operieren die meisten QA–Systeme auf Paragrafen von Texten, anstatt auf ganzen Texten. Daher wird meist die natürlichsprachliche Frage zuerst in eine Anfrage eines IR–Systems überführt, das damit eine relevante (meist sehr kleine) Menge von Dokumenten bestimmt. Das QA–System kann dann gezielter die relevanten Textpassagen bestimmen, in denen es die Antworten vermutet. Falls das IR–System keine Texte findet, kann das QA–System die Anfrage reformulieren und damit das IR–System erneut ansteuern. Ähnliche Interaktionsmuster lassen sich auch zwischen IE und IR modellieren, wie aktuelle Arbeiten im Bereich der domänenoffenen IE zeigen, vgl. Abschn.1.3.5.

1.2.1 Volltextsuche

Die zentrale Datenstruktur bei der Volltextsuche ist der **invertierte Index**, der eine Zuordnung zwischen Schlüsselwörtern und Dokumenten bereitstellt. Der invertierte Index fungiert als fundamentale Schnittstelle zwischen den Benutzern und den Texten, indem Anfragen als Schlüsselwörter interpretiert, auf den Index abgebildet und gefundene Dokumente geordnet nach ihrer Relevanz als Ergebnis geliefert werden. **Indexierung** nennt man den Prozess, der ein Vokabular aus Schlüsselwörtern mit allen Dokumenten eines Textkorpus (z. B. das gesamte Web) assoziiert. Meist wird eine **automatische Indexierung** durchgeführt, wobei beim Einlesen des Korpus aus jedem Dokument alle relevanten Wörter extrahiert werden. In der Regel sind dies die Inhaltswörter bzw. alle Wörter, die nicht in einer sogenannten **Stoppwortliste** vorkommen. Stoppwörter sind demnach solche Wörter, die bei einer Volltextsuche nicht beachtet werden sollen, da sie als inhaltlich irrelevant betrachtet werden und zu redundant sind, z. B. alle Wörter der geschlossenen Wortarten, wie Artikel oder Präposition. Das Sammeln (auch als **Crawling** bekannt) und **Einlesen sehr großer Korpora** (die Überführung von Folgen von Bytes in Folgen von Tokens) ist ein sehr komplexer Prozess an sich, da er nicht nur die verschiedenen Dokumentenformate und –kodierungen berücksichtigen muss, sondern auch die Granularität der Dokumenteneinheiten festlegt (z. B. ganze Dokumente, Paragrafen, Sätze), vgl. Manning et al. [2008] für mehr Details. Des Weiteren werden zur Verarbeitung sehr großer Korpora (insbesondere bei webbasierten Suchmaschinen) verteilte Indexierungsverfahren eingesetzt, wie z. B. das MapReduce Verfahren, das eine allgemeine Architektur für das verteilte Rechnen auf großen Clustern von Rechnern bereitstellt. Auf all diese Aspekte können wir leider nicht im Detail eingehen, vgl. aber Belew [2000] und Manning et al. [2008].

Jeder **Eintrag im invertierten Index** ist selbst eine komplexe Datenstruktur, die neben dem Schlüsselwort eine Liste aller Dokumente enthält (genauer aller $docID$s, von denen jede einen Zeiger auf ein Dokument repräsentiert), aus denen das Wort extrahiert wurde. Jede $docID$ kodiert auch jede Position

der Vorkommen eines Wortes im Dokument. Jeder Indexeintrag speichert daneben noch zusätzliche Informationen, wie die Häufigkeit des Wortes t (oft auch als **Term** bezeichnet)[2] pro Dokument d (Termfrequenz $tf_{t,d}$) und im gesamten Korpus c (Korpusfrequenz cf_t), die Anzahl der Dokumente, in denen der Term t vorkommt (Dokumentenfrequenz df_t) oder auch eine Liste von Zeigern auf Synonyme des Termes. Mit Hilfe dieser statistischen Information kann die Trefferliste eines Schlüsselwortes (die mit dem Term assoziierten Dokumente) nach der jeweiligen Relevanz des Termes innerhalb der Trefferliste sortiert werden.[3]

Ein beliebtes aus diesen Zahlen berechnetes Kriterium ist der tf-idf–Wert, der das Verhältnis zwischen der **Termfrequenz** (tf_t) von t und seiner **inversen Dokumentenfrequenz** (idf_t, das dem skalierten df_t entspricht, genauer $idf_t = \log \frac{|c|}{df_t}$) repräsentiert. Hierbei geht man davon aus, dass ein Wort umso charakteristischer für einen Text d ist, je weniger es in verschiedenen Texten vorkommt und je häufiger es in d vorkommt als in anderen. Anders ausgedrückt, unterschiedliche Wörter in einem Dokument d mit unterschiedlichem tf-idf–Werten müssen unterschiedlich signifikant für d sein.

Man kann sich leicht vorstellen, dass ein invertierter Index ebenfalls sehr groß werden kann, insbesondere, wenn alle Wortformen als unabhängige Einträge betrachtet werden. Daher findet bei der Indexerstellung oftmals auch eine **Normalisierung** der Wörter auf kanonische Formen statt, z. B. durch eine **Stammformbildung**. Weit verbreitet ist der Einsatz von so genannten **Stemming**-Verfahren, die meist eine lexikonfreie Suffixanalyse vornehmen und einfach an viele Sprachen anzupassen sind. Man beachte, dass Stemming quasi eine Komprimierung mit Verlust darstellt, da ja durch das Stemming Wörter mit unterschiedlichen Suffixen auf den selben Teilstring reduziert werden können, z. B. 'heizung', 'heizen', 'heizer', 'heizten' auf 'heiz'. Da auch die Schlüsselwörter aus der Anfrage entsprechend normalisiert werden müssen, verursacht Stemming im Allgemeinen eine erhöhte Ungenauigkeit in der Volltextsuche. Dies ist auch bekannt als **Overstemming** (semantisch unterschiedliche Wortformen werden auf denselben Stamm abgebildet, z. B. 'Wand' und 'Wandere' auf 'Wand') und **Understemming** (unterschiedliche Wortformen eines Wortes werden auf unterschiedliche Stämme abgebildet, z. B. 'absorbieren' und 'Absorption' auf 'absorb' und 'absorp'). Um diese Probleme zu vermeiden, werden auch vollständige morphologische Analysekomponenten eingesetzt, was allerdings die Sprachabhängigkeit erheblich erhöhen kann, vgl. Manning et al. [2008].

Der invertierte Index ist der zentrale Zugang zu den mit ihm assoziierten Dokumenten. Das bedeutet, dass **Suchanfragen** als Ausdrücke über Schlüsselwörtern interpretiert werden. Das hat u. a. den enormen Vorteil, dass die Formulierung von Suchanfragen sprichwörtlich kinderleicht ist. Im einfachsten Fall besteht eine Suchanfrage gerade aus einem Schlüsselwort, sehr oft nur aus einer Menge von zwei bis drei Schlüsselwörtern (das haben Analysen von Suchmaschinen–Logs gezeigt). Auf der anderen Seite bedeutet dies aber auch, das

[2] Wir werden im Weiteren daher beide Begriffe synonym verwenden.

[3] Zusätzlich kann auch noch andere Information zur Sortierung der Liste herangezogen werden, wie z. B. die Verlinkungsdichte eines Dokuments. Allerdings können wir auf diese Punkte nicht weiter eingehen, aber vgl. Manning et al. [2008].

ein Benutzer scheinbar eine sehr hohe Treffergenauigkeit von seinen gewählten Schlüsselwörtern erwartet, d. h. die seiner Meinung nach relevanten Dokumente sollen auch ganz oben in der Trefferliste erscheinen. Für die Entwickler von IR–Systemen hat dies zur Folge, dass die Vorteile einer einfachen Mensch–Maschine–Schnittstelle teilweise dadurch erkauft werden, dass die eigentlichen Suchprozesse entsprechend komplexer zu gestalten sind (die Entwicklung von modernen QA–Systemen ist auch darin motiviert, hier eine alternative Modellierung anzubieten, vgl. Abschnitt 1.4.1). Die Kunst besteht im Prinzip darin, einer kleinen Suchanfrage eine maximale Precision und möglichst hohen Recall zu entlocken.

Es ist hilfreich, für die weiteren Erläuterungen des Suchprozesses ein **Vektorraummodell** für den invertierten Index zu adaptieren. Im Vektorraummodell werden alle n Dokumente und alle m Schlüsselwörter in einer $n \times m$ Matrix angeordnet, wobei jedes Feld dt_{ij} entweder den Wert 0 hat (der Term t_j tritt nicht im Dokument d_i auf) oder einen Zahlenwert enthält, der der Relevanz des Terms im Dokument entspricht, z. B. gemessen durch dessen $tf\text{-}idf$–Wert. Die entsprechende Index Matrix sieht wie folgt aus:

$$\begin{pmatrix} dt_{11} & dt_{12} & dt_{13} & \cdots & dt_{1m} \\ dt_{21} & dt_{22} & dt_{23} & \cdots & dt_{2m} \\ dt_{31} & dt_{32} & dt_{33} & \cdots & dt_{3m} \\ \vdots & & & & \\ dt_{n1} & dt_{n2} & dt_{n3} & \cdots & dt_{nm} \end{pmatrix}$$

In dieser Darstellung wird also jedes Dokument als Vektor der Länge m kodiert, wobei jede Zelle kodiert, ob der Term (gewichtet) vorkommt oder nicht. Wenn man jeden Term als Merkmal betrachtet, dann entspricht jedes Dokument einem **Merkmalsvektor**. Ein Termvektor repräsentiert das (gewichtete) Vorkommen eines Termes in allen Dokumenten. Eine Anfrage (die ja auch aus Termen besteht) kann man ebenfalls in Form eines Dokumentenvektors kodieren, den wir als Q bezeichnen wollen und der obigen Matrix als $n+1$ Zeile hinzufügt werden kann.

Mit dieser Darstellung kann die Frage, welche Dokumente am besten zu einer Anfrage passen unmittelbar beantwortet werden: Es sind die Dokumente, die der Anfrage relativ zu einer gegebenen Metrik (der Gewichtung) am ähnlichsten sind. Das Vektorraummodell bietet daher den Vorteil, dass man eine uniforme Repräsentationsebene für Dokumente und Anfragen besitzt, mit der verschiedene **Ähnlichkeitsfunktionen** untersucht und verglichen werden können. Grundlage für viele Ähnlichkeitsfunktionen bildet der **Cosinus**. Hierbei wird die Matrix als m–dimensionaler Vektorraum interpretiert, wobei jedes Dokument einem Punkt entspricht. Da eine Anfrage Q auch als Dokument aufgefasst wird, entspricht Q ebenfalls einem Punkt. Nun ist es möglich, den Abstand zwischen Q und allen Dokumenten als Kriterum für Ähnlichkeit heranzuziehen: Je kleiner der Abstand ist, desto ähnlicher sind sich Anfrage und Dokument. Der Cosinus ist nun genau das Mittel, diese Abstände zu berechnen, vgl. auch Belew [2000].

Man beachte, dass eine Volltextsuche eine im Wesentlichen wortzentrierte Operation ist, auch wenn eine Anfrage aus mehreren Termen besteht, da die einzelnen Terme als unabhängig voneinander betrachtet werden. Das hat zwar den Vorteil, dass die Suche sehr flexibel und robust ist, hat aber auch den Nachteil, dass für kleine Anfragen die Suche im Prinzip sehr unterspezifiziert und unscharf ist. Manning et al. [2008] beschreiben eine Reihe von Erweiterungen, wie der Suchprozess kontrollierter realisiert werden kann, z. B. **Query Expansion** (beispielsweise durch das Hinzufügen von synonymen Wörtern) oder **Relevance Feedback** (der Benutzer bewertet explizit die Suchergebnisse, auf deren Basis das IR–System seine Suchstrategien zu optimieren versucht). QA–Systeme, speziell die Frageanalysekomponenten, versuchen ebenfalls gezielt eine Volltextsuche zu integrieren, da ausgehend von der syntaktischen und semantischen Analyse einer W–Frage ("Wer war in den 80iger Jahren Deutschlands Bundeskanzler?") gezielt ein IR–System angesteuert werden muss, um die relevanten Dokumente und Textpassagen für die Antwortextraktion zu bestimmen, vgl. Abschn. 1.4.1.

1.2.2 Verzeichnisbasierte Suche

Eine weitere Möglichkeit ist es, die Dokumentenmenge in Teilmengen zu zerlegen, umso eine Art begriffliche Strukturierung der Dokumente in thematisch ähnliche Verzeichnisse zu erhalten. Zum Beispiel könnte man die Verzeichnisse "Umsatz von Firmen", "Computer", "Unterhaltung" und "Sonstiges" definieren und die Dokumente eines Korpus entsprechend zuordnen. Im Prinzip könnte man auch pro Verzeichnis weitere Unterverzeichnisse festlegen, etwa "Umsatz von Firmen in der Computerbranche" oder "Umsatz von Firmen im Bankenbereich". Dann kann man einen zweistufigen Suchprozess anbieten, wo in der ersten Stufe der Benutzer zuerst das Verzeichnis auswählt (dies entspricht im Prinzip einem Navigieren in einer Hierarchie). Je nach Thema bekommt man dann entweder alle dem gerade ausgewählten Verzeichnis zugeordneten Unterverzeichnisse oder eine Liste von aktuellen Dokumenten angeboten, in denen man dann wie gewohnt durch Angabe von Schlüsselwörtern suchen kann, wobei natürlich nur in der relevanten Teilmenge der Dokumente gesucht wird. Beispiele für solche verzeichnisbasierten Suchmaschinen sind das Google Verzeichnis (http://www.google.de/dirhp?hl=de) oder das Open Directory Project (http://www.dmoz.de/).

Theoretisch ist es möglich, die Verzeichnisstruktur vorzugeben (z. B. durch ein Thesaurus oder eine Ontologie, vgl. **??**) und Dokumente manuell zuzuordnen. Allerdings ist dies ein zeitaufwändiger und sehr kostenintensiver Prozess, insbesondere für sehr große Dokumentensammlungen. Daher werden gerade in diesem Bereich meist maschinelle Lernverfahren eingesetzt, die wir hier kurz vorstellen wollen.

1.2.3 Textklassifikation

Im Prinzip kann man den Bezeichner eines Verzeichnisses auch als eine Art "eingefrorene Suchanfrage" (**standing query**) auffassen, im Gegensatz zu den bisher betrachteten "spontanen Suchanfragen" (**ad hoc queries**). Bei den ad hoc queries werden die Suchergebnisse quasi vergessen, wenn der Benutzer eine neue Anfrage stellt. Dadurch wird es aber sehr schwierig, Veränderungen in der Dokumentenmenge zu registrieren, wenn die Frage zu einem späteren Zeitpunkt erneut gestellt wird (z. B. für die Anfrage "aktuelle Preise von Blue Ray Recordern"). Die Idee der standing queries ist es nun, diese Anfragen periodisch auszuführen und die gefundene relevante Dokumentenmenge inkrementell zu erweitern, d. h., die mit einer standing query gespeicherten (alten) Dokumentenmenge zu aktualisieren.

Dokumente, die als Ergebnis einer standing query Q bestimmt werden, sind Texte über Q. Daher können standing queries auch als Klassenbezeichner aufgefasst werden, die die Dokumentenmenge in solche zerlegen, die *über* Q sind und die *nicht über* Q sind. Damit kann die Beantwortung von standing queries auch als **Textklassifikation** betrachtet werden: Gegeben sei eine Menge von Klassen. Bestimme für jedes Dokument, zu welcher Klasse es gehört. Anstelle von **Klasse** wird auch oft der Term **Topik** verwendet und Textklassifikation dann auch als **Topik Klassifikation** oder **Topik Spotting**.

Weitverbreitet, weil sehr erfolgreich, sind Statistik-basierte Verfahren zur Textklassifikation. Diese Verfahren lernen automatisch die Entscheidungskriterien aus einer Menge von Trainingsbeispielen, d. s. gute Beispieltexte für jede Klasse. Sie werden daher auch als **überwachte** Lernverfahren bezeichnet. Das Klassifizieren der Beispiele wird zwar in der Regel manuell durchgeführt, ist aber relativ einfach (man muss kein Wirtschaftsexperte sein, um zu entscheiden ob ein Text eine Umsatzmeldung beschreibt oder nicht). Oft ist der Prozess der Annotation auch Teil des Informationsflusses, wenn z. B. neue Dokumente, die mit einer standing query bestimmt wurden, als relevant oder nicht relevant für eine Klasse zu klassifizieren sind. Aufgabe der Lernverfahren ist es, mit den Trainingsdaten eine Wahrscheinlichkeitsverteilung zu berechnen, mit deren Hilfe die beste Klasse für neue Dokumente bestimmt werden kann. Dabei wird für jede mögliche Klasse der sich aus der Verteilung ergebende Wert berechnet und die Klasse mit höchstem Wert gewählt. Die Verteilungsfunktionen operieren über Merkmalen, wie z. B. den Wörtern der Texte, Wortfolgen, Wortarten etc. Daher werden die Texte entsprechend vorbearbeitet und in einem **Merkmalsvektor** auf ähnliche Weise repräsentiert, wie wir dies bereits in Abschn. 1.2.1 bei der Volltextsuche beschrieben haben.

Zu den besten Textklassifikatoren (F-Maß von mehr als 88%)[4] gehören die stützmengenbasierten Verfahren, die so genannten **Support Vector Machines** (SVM), vgl. Schölkopf and Smola [2002]. Für gegebene positive und negative Beispiele (da dies Vektoren sind, also Punkte im Vektorraum) ist es das Ziel, eine lineare Funktion zu bestimmen, die die Punkte dieser Klassen optimal voneinander trennt, wobei nur die Punkte (sog. **Stützpunkte**) betrachtet wer-

[4] Vgl. Li and Yang [2003]

den müssen, die zur Funktion den geringsten Abstand haben. Optimal bedeutet, dass in der direkten Umgebung der Funktion ein möglichst großer Bereich ist, in dem keine Punkte vorkommen. Diese Eigenschaft erlaubt, neue Texte recht zuverlässig zu klassifizieren. SVM–Verfahren gehören zu den Kernel–Verfahren, die wir in Abschn. 1.3.3 im Kontext der Extraktion von Relationen kurz erläutern werden. Weitere Details zu SVM–basierten Textklassifikatoren und zu anderen Verfahren, wie etwa **Naive Bayes**, **Rocchio** oder **k–Nearest–Neighbor** (k–NN) finden sich in Manning et al. [2008].

Ein mögliches Anwendungsfeld der Textklassifikation ist die Informationsextraktion (IE), wo auf diesem Wege relevante Texte periodisch bestimmt werden können, aus denen dann die wichtigen Entitäten und Relationen extrahiert werden (wie wir dies ja bereits im Beispiel in Abschn. 1.1.3 angedeutet haben). Der Vorteil ist hierbei, dass die IE direkt auf die so gefilterten Dokumente angewendet werden kann und sich somit nicht um "falsche" Texte kümmern muss, vgl. Abschn. 1.3. Weitere Anwendungsfelder sind u. a. das automatische Filtern oder Weiterleiten von Dokumenten, das Bestimmen von Spam–Seiten, von Texten mit pornografischen Inhalten oder mit subjektiven positiven oder negativen Bewertungen von Produkten (auch als Sentiment Detection bekannt) oder zur Sortierung von Emails.

1.2.4 Textclustering

Clustering ist die Verteilung von Daten in Gruppen von ähnlichen Objekten und leistet damit auch ein automatisches Erstellen von Klassen. Jede Gruppe wird als Cluster bezeichnet und besteht aus den Objekten, die einerseits untereinander ähnlich sind und andererseits unähnlich zu den Objekten der anderen Gruppen. Aus Sicht eines TIMs meint Ähnlichkeit hier, dass sich in einer Gruppe möglichst nur die Objekte befinden sollen, die bezogen auf ein Informationsbedürfnis die gleiche Relevanz haben. Für ein IR–System sind dies z. B. die Dokumente, die für eine Suchanfrage relevant sind; für ein IE–System mögen dies Textpassagen sein, die die gleichen Eigennamen und Relationen enthalten und für ein QA–System könnten dies Sätze aus verschiedenen Dokumenten sein, die Formulierungsvarianten derselben Antwort enthalten.

Clustering–Verfahren sind **unüberwachte** Lernverfahren, da sie keine Trainingsbeispiele haben. Auch sind die Gruppenbezeichner nicht explizit bekannt. In diesem Sinne entspricht ein Cluster einem "verstecktem" (**hidden**) Muster. Im Falle eines TIMs entsprechen die Objekte linguistisch Objekten, wie etwa Texten, Sätzen, Wörtern oder Wortpaaren, oder auch Relationen. Die Mehrzahl der Clustering–Verfahren modelliert Objekte ebenfalls als Punkte–zu–Merkmale Datenformate, die konzeptuell ebenfalls einer $n \times m$-Matrix entsprechen (dem Merkmalsraum für n Objekte und m verschiedene Merkmale). Das Ziel des Clusterings ist es dann, die Punkte einer Menge X einem endlichen System von k Teilmengen, den Clustern C_i, zuzuweisen, sodass die einzelnen Cluster sich möglichst nicht schneiden, d. h. $X = C_1 \cup C_2 \ldots C_k \cup C_{outliers}$, wobei $C_i \cap C_j = \emptyset$. Die Ähnlichkeit von Objekten kann dann z. B. mittels eines Distanzmaßes zwischen entsprechenden Punkten bestimmt werden, sodass der "Abstand" von

Texten innerhalb eines Clusters möglichst minimal und die Distanz zwischen verschiedenen Clustern möglichst maximal ist.

Methoden zum Clustering können grob aufgeteilt werden in hierarchische und flache Verfahren. Beim **hierarchischen Clustering** ergibt der Clusteringprozess eine Baumstruktur. Jeder Clusterknoten enthält Kindercluster, wogegen Geschwistercluster die Punkte des Elternclusters partitionieren, vgl. Berkhin [2006]. Methoden zum hierarchischen Clustering können weiter unterteilt werden in bottom–up (**agglomerative**) und top–down (**divisive**). **Agglomeratives Clustering** beginnt mit Clustern, die jeweils nur einen Punkt (Text) enthalten und vereinigen rekursiv die jeweils zwei aktuell geeignetsten Cluster zu neuen Clustern. **Divisives Clustering** startet mit einem Cluster, der alle Punkte enthält und spaltet rekursiv das aktuell beste Cluster ab, solange das Terminierungskriterium nicht erreicht ist (oft ist dies eine vorgegebenen Anzahl k von Clustern). Die wichtigsten Vorteile des hierarchischen Clusterings sind 1) die Granularität der Clusterhierarchie kann gesteuert werden, 2) verschiedenste Distanzmaße können integriert werden und 3) sind damit beliebige Merkmalstypen wählbar. Ein gravierender Nachteil ist das vage Terminierungskriterium, da die Vorgabe von k schon einen erheblichen Eingriff in das Clustering–Verfahren bedeutet und oftmals nur durch Schätzung oder mit Hilfe von Domänenwissen bestimmt werden kann.

Flaches Clustering erzeugt eine flache Clusteringstruktur, d. h. es gibt keine explizite strukturelle Beziehung zwischen den Clustern. Sehr populär in wissenschaftlichen und industriellen Anwendungen ist der **K–means** Algorithmus. Jedes der k Cluster C_i wird durch den Durchschnittswert c_i seiner Punkte definiert. c_i heißt dann auch **Zentroid**. Ziel ist es, die durchschnittliche Distanz zwischen Texten und ihren Zentrodien zu minimieren bzw. die Ähnlichkeit zwischen Texten und ihren Zentroiden zu maximieren. Zuerst werden zufällig k Punkte als initiale Zentroide bestimmt und von diesen ausgehend die anderen Punkte um die Zentroide angeordnet. Für die sich so ergebenen Cluster werden dann die Zentroiden neu bestimmt. Auf dieser Basis wird der Prozess rekursiv solange über den k Clustern fortgeführt, bis keine Veränderungen bzw. Bewegungen der Zentroiden mehr stattfinden. K–means Verfahren sind einfach und schnell, haben aber den Nachteil, dass sie nur numerische Merkmale für die Distanzberechnung verwenden können. Außerdem reagieren sie sehr fragil auf die initiale Auswahl von Zentroiden.

Wie erwähnt wird Clustering bereits aktuell zur dynamischen Erzeugung von Verzeichnissen in Suchmaschinen eingesetzt, vgl. z. B. http://clusty.com/. Manning et al. [2008] beschreiben weitere Einsatzgebiete des Clusterings im IR. In Forschungsprototypen von QA–Systemen wird Clustering oft zur Erkennung von Paraphrasen eingesetzt und zur Gruppierung von semantisch ähnlichen Textpassagen, womit eine gezielte Antwortextraktion möglich wird, vgl. Abschn. 1.4.1. Aktuell werden Clustering Verfahren verstärkt im Bereich der IE exploriert, insbesondere im Bereich der Behandlung mehrdeutiger Eigennamen (vgl. Abschn. 1.3.2) und der webbasierten, domänenoffenen IE, vgl. 1.3.5.

1.3 Informationsextraktion

Das Ziel der Informationextraktion (IE) ist es, domänenspezifische Informationen aus freien Texten gezielt aufzuspüren und zu strukturieren, bei gleichzeitigem "Überlesen" irrelevanter Information. Die IE–Technologien versuchen keine umfassende Analyse des gesamten Inhaltes aller Textdokumente, sondern sollen nur die Textpassagen analysieren bzw. "verstehen", die relevante Information beinhalten. Was als relevant gilt, wird dabei durch vordefinierte domänenspezifische Lexikoneinträge oder Regeln dem System fest vorgegeben. Dieses Wissen muss dabei so detailliert und genau wie möglich festlegen, welche Typen von Information von einem IE-System extrahiert werden soll, damit eine umfangreiche und zugleich präzise Extraktion ermöglicht wird. Dieser letzte Aspekt betrifft zumindest die **traditonelle IE**, die wir in diesem Abschnitt genauer beleuchten werden. In aktuellen modernen IE–Ansätzen — insbesondere **domänenoffene IE** — wird versucht, alle möglichen relevanten Informationen in einer Textkollektion zu erkennen und zu klassifizieren. Diese Verfahren werden wir in Abschnitt 1.3.5 genauer erläutern.

Beiden IE–Ansätzen gemeinsam ist das Ziel, in freien Texten das textuelle Vorkommen (die Erwähnung) von Entitäten und ihre Beziehungen zueinander zu lokalisieren und in ein strukturiertes Format zu überführen. Damit streben sie eine tiefere Verstehensleistung an, als die volltextbasierte IR. IE kann so betrachtet auch als eine Art "schärferes Textzooming" betrachtet werden und damit als ein dem IR nachfolgender Analyseprozess. Die wichtigsten konkreten Lösungsschritte umfassen 1) die Erkennung von spezifischen Eigennamen (z. B. Namen von Personen, Firmen, Produkten, Lokationen), 2) die Referenzierung der erkannten Eigennamen (z. B. bestimmen, dass "Angela Merkel", "A. Merkel" und "die Bundeskanzlerin" dieselbe Person referieren) und 3) die Erkennung und Klassifikation von Relationen unterschiedlicher Komplexitäten zwischen den identifizierten Eigennamen. Mit anderen Worten ist die zentrale Aufgabe der Informationsextraktion die Extraktion von **semantischen Relationen**. Beispielsweise könnte aus folgendem Text

> Die Hauptgeschäftsstelle von Apple Inc. befindet sich in der Mitte von Silicon Valley, in 1 Infinite Loop, Cupertino, Kalifornien.

die Relation HATHAUPTGESCHÄFTSSTELLE(APPLE INC., 1 INFINITE LOOP–CUPERTINO–CALIFORNIA) extrahiert werden. Dabei ist die zentrale Herausforderung diese Relation aus Sätzen zu extrahieren, die die selbe Information ausdrücken unabhängig von der konkreten sprachlichen Formulierung. Desweiteren sollten auch entsprechende Informationen über andere Firmen extrahiert werden können, wenn diese in ähnlich formulierten Texten vorkommen, also dieselbe Relation verbalisiert ist, aber mit anderen Argumenten, z. B. HATHAUPTGESCHÄFTSSTELLE(IBM CORP., 1 NEW ORCHARD ROAD–ARMONK–NEW YORK).

Die frühe Generation von IE–Systemen bestand im Wesentlichen aus manuell implementierten Systemen, in denen der notwendige domänenspezifische Regelapparat für alle Teilbereiche per Hand programmiert wurde, vgl. u. a. Is-

rael et al. [1996] oder Neumann et al. [1997] und Carstensen [2008] für einen genaueren geschichtlichen Einblick in die Entwicklung von IE–Systemen. Unser Beispiel von oben aufgreifend bedeutet dies, dass für eine bekannte semantische Relation HATHAUPTGESCHÄFTSSTELLE eine Menge von Regeln oder Muster der Art (hier sehr vereinfacht):

* HAUPTGESCHÄFTSSTELLE VON X BEFINDET SICH * IN Y

manuell zu erstellen ist, wobei X und Y getypte Variablen der Argumente (in der Regel Eigennamen) repräsentieren. Dieses Muster passt auf alle Textstellen, wo auf eine beliebig lange Folge von Termen (was durch den Kleene–Stern "*" kodiert ist) die Wörter "Hauptgeschäftsstelle von" genau in dieser Form und Anordnung folgen, denen sich unmittelbar ein Eigenname vom Typ X anschließt, dem wiederum die Sequenz "befindet sich" folgt und dieser schließlich ein Text folgt, der auf analoge Weise mit dem Teilausdruck * IN Y passend ist.

In manuellen IE–Systemen wird die Spezifikation solcher Muster auf der Basis einer manuellen Analyse relevanter Textkorpora geleistet, wobei diese jedoch oft durch eine automatische linguistische Merkmalsextraktion aktiv unterstützt wird. Die manuelle Durchsicht eines Korpus hat zwar den Vorteil, dass zum Einen sehr genaue Muster bestimmt werden können und zum Anderen auch die relevanten linguistischen Merkmale eindeutig identifiziert werden können. Ein gravierender Nachteil ist, dass dieser Prozess sehr zeit– und kostenintensiv ist und dadurch oft nur eine geringe Abdeckung mit manuellen System erreicht wird, insbesondere dann, wenn eine IE–Anwendung zeitnah bereitzustellen ist.

Daher basieren die aktuell am weitesten fortgeschrittenen Technologien algorithmisch auf Verfahren des Maschinellen Lernens, die unterschiedliche Granularitäten von linguistischer Merkmalsextraktion in Betracht ziehen, z. B. von POS-Tagging bis zu vollem Parsing. Ziel dieser Lernverfahren ist es, durch automatische Bestimmung optimaler Kombinationen von Merkmalen die Extraktionsmuster zu induzieren, die später in der Anwendungsphase zur automatischen Erkennung und Klassifikation von Entitäten und Relationen herangezogen werden. Damit kann dann eine sehr viel größere Abdeckung erreicht werden, wobei die aktuell besten Strategien bereits eine vielversprechende Genauigkeit demonstrieren. Die zugrunde liegenden Lernstrategien der Mehrzahl der neuen Ansätze reichen von überwachten über semi-überwachten bis zu unüberwachten Lernmethoden mit einer aktuell starken Tendenz zu semi-überwachten und unüberwachten Methoden. Wir werden nun in den nächsten zwei Abschnitten zuerst etwas genauer auf die Extraktion von Eigennamen und Relationen eingehen, bevor in Abschnitt 1.3.2 Strategien zu deren Dereferenzierung und Integration vorgestellt werden.

1.3.1 Extraktion von Eigennamen

Die Erkennung und Klassifikation von Eigennamen stellt einen zentralen Baustein für ein TIM dar. Unter Eigennamen versteht man in der Regel sprachliche Ausdrücke, die auf Individuen von Klassen oder Typen bestimmter Entitäten

referieren, wie z. B. wie Personen-, Firmen-, Produktnamen, komplexe Datums-, Zeit-, und Maßausdrücke.

Die einfachste Art der Eigennamenerkennung (**Named Entity (NE) Recognition, NER**) wäre sicherlich die vollständige Aufzählung aller Eigennamen eines bestimmten Types in einer Liste (die dann auch als **Gazetteer** bezeichnet wird), z. B. eine Liste aller Personennamen. Dies ist sicherlich ein gangbarer erster Implementationsweg (und ist auch weitverbreitet), aber sicherlich kein ausreichender. Zum Einen gibt es für einen Eigennamen in der Regel viele Formulierungsmöglichkeiten (z. B. "Angela Merkel", "A. Merkel", "Merkel, Angela", "Merkel, A.", "Angie", "Merkel, Dr. Angela", "A. M.", etc.), zum Anderen ist der Prozess der Generierung von Eigennamen sehr produktiv, z. B. bei Firmennamen oder Produktnamen. Hinzu kommt, dass durch die immer noch stattfindende Expansion von IE Anwendungen und das Aufkommen von neuen verwandten Themen (wie etwa domänenoffene Fragebeantwortung, semantische Suche), auch eine drastische Erweiterung der NE–Kategorien stattgefunden hat. So stellt z. B. Sekine et al. [2002] ein System bestehend aus rund 150 verschiedenen Kategorien vor.

Auf Grund dieser Eigenschaften und Entwicklungen dominiert heute die Entwicklung von Daten–gesteuerten Verfahren zur NER mit einem Fokus auf überwachte und semi-überwachte Strategien. Die Idee **überwachter Lernverfahren** ist die automatische Berechnung von optimalen Merkmalskombinationen aus positiven und negativen Beispielen von NEs, die in einem großen Textkorpus in der Regel manuell annotiert vorliegen. Diese Merkmalskombinationen dienen dann als Vorhersagemodelle zur **Erkennung und Klassifikation** von neuen Vorkommen von NEs. Die Merkmale beziehen sich dabei zum Einen auf die Schreibweise von Eigennamen (z. B. Großschreibung, Vorkommen von invariablen Designatoren, wie etwa AG oder Inc. oder die Länge der Buchstabenfolge) und zum Anderen auf Information im Kontext von NE, z. B. Wortform oder Wortart adjazenter linker und rechter Wörter oder aber auch komplexe syntaktische Merkmale, wie Vorkommen eines NEs in einer bestimmten syntaktischen Konstruktion einer Nominalphrase. Diese Lernverfahren sind oft Anpassungen von bekannten Lernverfahren, wie Hidden–Markov Modellen (HMM, Bikel et al. [1999]), Maximum Entropy Modellen (MEM, Borthwick et al. [1998]), Support Vector Machines (SVM, Asahara and Matsumoto [2003]) oder Conditional Random Fields (CRF, Li and McCallum [2003]) oder auch Kombinationen von verschiedenen Verfahren, z. B. MEM und datengesteuertes Parsing (Neumann [2006]).

Der zentrale Nachteil überwachter Lernverfahren ist, dass sie sehr große annotierte Korpora zum Trainieren benötigen, insbesondere dann, wenn sehr viele NE–Kategorien vorliegen. Daher werden auch alternativ sogenannte **semi–überwachte Verfahren** erforscht und sehr aktuell unüberwachte Lernverfahren (auf die wir in Abschnitt 1.3.5 genauer eingehen werden). Die zentrale Technologie von semi–überwachten Verfahren ist als **Bootstrapping** bekannt. Diese Verfahren verfügen über einen sehr kleinen Grad an Überwachung in Form einer kleinen Menge von bereits bekannten Eigennamen (auch als **Seeds** bezeichnet) pro zu erlernender NE–Kategorie, die zur Initialisierung des Lernvorgangs her-

angezogen werden. Wenn es beispielsweise das Ziel ist, Namen von Krankheiten in einem Textkorpus zu bestimmen, dann könnte der NE–Lerner mit einer Liste von fünf bekannten Namen von Krankheiten gestartet werden. Der NE–Lerner sucht dann das gesamte Textkorpus nach Vorkommen dieser Seeds ab und annotiert die entsprechenden Textstellen mit dem Namenstyp. In einem nächsten Schritt werden dann automatisch die besten oder sichersten Merkmalskombinationen bestimmt, die sich im Kontext (und der Schreibweise) der verschiedenen Vorkommen der Seeds bestimmen lassen. Dies ergibt eine erste Menge von Mustern. In einem nächsten Schritt werden diese Muster herangezogen, um nun ihrerseits neue Eigennamen zu bestimmen, die der Seedliste (nachdem die einzelnen Einträge vom NE–Lerner selbstständig bewertet wurden) hinzugefügt werden. Dieser Prozess wird dann solange wiederholt, bis keine neuen Einträge mehr gefunden werden können oder der Prozess von außen terminiert wird (siehe auch Nadeau and Sekine [2007]). Beispiele für solche Lernverfahren finden sich u. a. in Collins and Singer [1999] und Yangarber et al. [2002], wobei sich die Verfahren auch in der Tiefe der linguistischen Merkmalsextraktion unterscheiden. So verlangt z. B. das Verfahren von Collins, dass das gesamte Korpus vollständig geparst wird, bevor der Lernprozess gestartet werden kann, wogegen Yangarbers Methodik nur ein POS–Tagging benötigt.

Zentrale Nachteile der aktuellen Verfahren sind zum Einen, dass sie ein sehr viel größeres Korpus benötigen als überwachte Verfahren und zum Anderen, dass die konkrete Auswahl der Seedelemente einen großen Einfluss auf die Performanz des NE–Lerners haben kann, z. B. wenn Namen gewählt werden, die z. B. nur sehr spärlich und in sehr seltenen Kontexten vorkommen, was man aber im Vorfeld der Anwendung eines semi–überwachten Lerners in der Regel nicht wissen kann (eine manuelle Sichtung des Korpus soll ja vermieden werden).

1.3.2 Mehrdeutige Eigennamen

Die Extraktion von Eigennamen im engen Sinne erkennt und klassifiziert einzelne Ausdrücke, leistet aber noch keine Auflösung von mehrdeutigen Eigennamen. Im allgemeinen handelt es sich hierbei um eine viele-zu-viele Relation, die durch zwei lexikalisch–semantische Relationen verursacht wird: 1) Polysemie: Derselbe Name verweist auf mehrere Entitäten und 2) Synonymie: Dieselbe Entität kann mehrere Namen haben. Ziel der **Disambiguierung von Eigennamen** ist es, diese Relation für die extrahierten Namen explizit zu berechnen.

Eine aktuell sehr im Fokus stehende Anwendung ist die **Personensuche im Web**, wofür es auch bereits erste spezialisierte Suchmaschinen gibt, z. B. http://www.spock.com/ oder http://www.zoominfo.com/. Meistens werden die zugrundeliegenden persönlichen Einträge und Profile zunächst von der jeweiligen Person selbst zur Verfügung gestellt. Es gibt aber bereits aktuelle Forschungsansätze diese Information voll automatisch durch die Analyse von Webseiten zu bestimmen. Dabei werden als Informationsquelle Webseiten inspiziert und klassifiziert, vgl. Artiles et al. [2007]. Der Informationsfluss ist in allen dort vorgestellten Systemen ähnlich. Zuerst wird der Textteil aus den HTML–Seiten extrahiert (basierend auf maschinellen Lernverfahren oder Regeln). Danach werden

die Texte mit einer Kombination aus Web und LT–Tools weiter analyisiert, u.a. mittels Stemmer, POS–Tagger, Chunker, NER, Koreferenzierer[5], sowie durch Extraktion von Fakten, Email und URL und Analyse der Links. Alle diese extrahierten Information werden dann als Merkmale für ein nachfolgendes Clustering der Webseiten eingesetzt, sodass ein Cluster ein Individuum repräsentiert. Dabei hat sich gezeigt, dass eine fehlerfreie Erkennung von Eigennamen und die Extraktion von Relationen auf der Basis von vollem Parsing von steigender Wichtigkeit ist, um feinkörnige Informationseinheiten lokalisieren und klassifizieren zu können, vgl. http://nlp.uned.es/weps/.

Verfahren zur Disambiguierung von Eigennamen verknüpfen Namen mit Entitäten, die in **externen Wissensquellen** vordefiniert sind, z. B. in Form von externen strukturierten Lexika, in denen die verschiedenen Entitäten mittels getypten Attributen unterschieden werden. Aktuell im Fokus der Forschung steht die Entwicklung von Verfahren, die diese externen Wissensquellen automatisch erstellen können. Von besonderem Interesse ist hierbei die Verwendung von Wikipedia (vgl. auch **??**), da **Wikipedia** tatsächlich als eine der größten verfügbaren Ressourcen dieser Art betrachtet werden kann. Man kann Wikipedia–Artikel über Entitäten (also Artikel, deren Titel im Wesentlichen aus einem Eigennamen besteht, z. B. "Jim Clark") als deren Definition betrachten. Dann ist es das Ziel, für Eigennamen, die aus beliebigen anderen Textquellen extrahiert wurden, die entsprechenden korrekten Wikipedia–Artikel zu bestimmen. Dies ist keine triviale Aufgabe, da die Eigennamen in der Regel normalisiert werden müssen (z. B. "J. Clark", "James Clark" auf "Jim Clark") und die Ähnlichkeit des Quelltextes mit dem Wikipedia–Artikel bestimmt werden muss, vgl. u.a. Bunescu and Pasca [2006] und Cucerzan [2007].

1.3.3 Extraktion von Relationen

Die zentrale Aufgabe im Bereich der **Relationsextraktion (RE)** ist das *Erkennen und Klassifizieren von Tupeln* von Entitäten, für die es textuelle Belegstellen gibt, dass diese Entitäten in einer bestimmten, in der Regel vordefinierten Relation zueinander stehen. Die textuellen Belegstellen entsprechen dabei Textausschnitten, die die Entitäten und die Relation direkt formulieren. Sucht man beispielsweise nach Tupeln (oder Instanzen) einer vordefinierten Relation vom Typ ÜbernahmeFirma(Firma1, Firma2), dann wäre der folgende Text eine positive Belegstelle für das Tupel ⟨**Google**, **YouTube**⟩:

> 10. Okt. 2006 ... Die Gerüchteküche irrte sich nicht: **Google** hat die Videoplattform **YouTube** übernommen. Der Kaufpreis liegt mit 1,65 Milliarden US-Dollar in ...

wogegen der folgende Text kein positiver Beleg für diese Relation ist:

[5] Das ist eine Komponente, die bestimmt, welche sprachlichen Ausdrücke (z. B. Eigennamen und Nominalphrasen) auf dieselben Entitäten verweisen und diese entsprechend verknüpft, vgl. auch Abschn. **??** in Kapitel **??**.

> 19. Nov. 2008 ... Das fragt Alex Iskold auf ReadWriteWeb. Er erzählt die Geschichte eines neunjährigen Jungen, welcher statt **Google** immer **YouTube** verwendet, ...

Diesem ersten Beispiel können wir bereits wichtige, häufig gemachte Annahmen entnehmen, die im Prinzip den meisten RE–Verfahren zu Grunde liegen. Erstens: Die Argumente der Zielrelation sind normalerweise Entitäten mit vordefiniertem Typ. In unserem Beispiel sind etwa nur Argumente vom Typ FIRMENNAME zugelassen. In den meisten Fällen sind die Argumente tatsächlich auf Eigennamen beschränkt (etwa PERSONEN, ORGANISATIONEN, ORTE, ZEITPUNKTE), da Relationen zwischen Eigennamen oft einen sehr hohen Informationswert (und Unterhaltungswert) besitzen. Zweitens: Die Relationsextraktion wird meist beschränkt auf das Auffinden einer vordefinierten Relation zwischen Entitäten, die im selben Satz auftreten. Drittens: Die RE–Aufgabe kann dann auch als Klassifikationsaufgabe für Sätze aufgefasst werden zu entscheiden, ob Sätze, die die gleichen (Typen von) Eigennamen nennen, auch die gleiche Relation benennen oder nicht.

Die ersten Maschinellen Lernverfahren für RE waren Varianten von **überwachten** induktiven Lernverfahren. Ausgehend von einer manuell erstellten Trainingsmenge von positiven und negativen Sätzen (in denen die Eigennamen natürlich auch markiert sind), ist es das Ziel, automatisch Regeln zu induzieren. Dazu werden schrittweise die durch die Trainingsmenge vorgegebenen Sätze verallgemeinert, sodass sie auch auf nichtannotierte, neue Dokumente möglichst fehlerfrei anwendbar sind. Auf diese Weise könnte zum Beispiel aus einer Menge von positiven Trainingssätzen, die Interaktionen zwischen Proteinen ausdrücken, folgende Regel abgeleitet werden (vgl. auch Bunescu et al. [2005]):

> (7) interaction (0) [between | of] (5) PROT (9) PRO (17) .

wobei die geklammerten Zahlen (z. B. (7)) ausdrücken, wie viele beliebige Tokens oder Wörter maximal zwischen den explizit genannten Wörtern (z. B. interaction) oder den Eigennamen (z. B. Eigennamen vom Typ PROT, also Protein) vorkommen dürfen. Wörter in eckigen Klammern und durch | getrennt repräsentieren alternative Formulierungsmöglichkeiten an diesen Stellen in Sätzen.

Um zu diesen **domänenspezifischen Regeln** zu gelangen, werden die Sätze mit verschiedenen linguistischen Werkzeugen vorverarbeitet, von reiner Wortanalyse (etwa Freitag [1998]), über POS–Tagging (etwa Califf and Mooney [1999]) und Phrasenerkennung (etwa Huffman [1996]) bis zu vollem Parsing (etwa Soderland [1997]). Die verwendeten Verfahren verlaufen meist bottom–up und vergleichen Sätze und Regeln wortweise, wobei die zugrundeliegende Suchstrategie oft Greedy (lokale Optima bevorzugend) ist. Die resultierenden Regeln können als kompaktifizierte Sequenzen von Wörtern (plus linguistischen Merkmalen) betrachtet werden, wobei die Wörter als explizite **lexikalische Anker** dienen. Das heißt, dass in neuen Sätzen genau an den gleichen Stellen die gleichen Wörter auftreten müssen, damit die Muster überhaupt angewendet werden können. Das bedeutet dann auch, dass Sätze, in denen keine solchen lexikalischen Anker vorkommen, direkt ausgefiltert (quasi "überlesen") werden können.

Damit die erlernten Regeln einen möglichst hohen Abdeckungsgrad haben, ist die Anzahl solcher lexikalischen Anker in einer Regel meist klein. Das führt in der Anwendungsphase häufig dazu, dass die erlernten Regeln zwar eine hohe Präzision besitzen, aber nur eine geringe Vollständigkeit (sie "überlesen" zu viel).

Neuere Lernverfahren zur RE versuchen dieses Problem dadurch zu beheben, dass sie nicht nur einen wortweisen Vergleich vornehmen, sondern dass sie alle mögliche Teilfolgen von Wörtern in Sätzen betrachten und es dem Trainingsalgorithmus überlassen, die besten Teilfolgen zu bestimmen. Prinzipiell lassen sich diese Teilfolgen automatisch vorberechnen und könnten somit als zusätzliche Merkmale benutzt werden. Allerdings führt solch eine explizite Erzeugung von Merkmalen zu einem enormen Anwachsen des expliziten Suchraums und damit zu nicht mehr machbaren Trainingszeiten. Aktuell erforschte RE–Lernverfahren versuchen daher, diese Berechnungen quasi implizit durch eine Funktion (genannt **Kernel**) zu bestimmen, die die Ähnlichkeit von Objekten (z. B. Sätzen) bestimmt. Je größer der Wert des Kernels, desto ähnlicher sind die verglichenen Objektpaare. Dabei liegt der Fokus nicht auf der expliziten Repräsentation von Objekten als Merkmalsvektor, sondern auf dem Datentyp der Objekte. Häufig verwendete Datentypen sind Zeichketten, Sequenzen beliebiger Elemente, Baumstrukturen oder sogar Graphen. Ein Kernel kann dann als rekursive Funktion modelliert werden, in denen sehr effiziente elementare Vergleichsoperatoren effizient zusammengefügt werden. So lässt sich z. B. die Ähnlichkeit von zwei Sequenzen durch die Anzahl der gleichen Teilsequenzen bestimmen oder die Ähnlichkeit von zwei Bäumen durch die Anzahl der gleichen Teilbäume.

Ein zentraler Vorteil von Kernels ist, dass sie sehr elegant mit maschinellen Lernverfahren verknüpft werden können, wie z. B. Support Vector Machines (SVM) oder aber auch Latent Semantic Analysis. Hierbei ist das Ziel, ausgehend von einer Menge von Trainingsbeispielen einen problemspezifischen optimalen Kernel zu bestimmen, vgl. z. B. Zelenko et al. [2003], Culotta and Sorensen [2004], Bunescu and Mooney [2005], Zhao and Grishman [2005], Bunescu [2007], Wang and Neumann [2007] und Moschitti et al. [2008].[6]

Die bisher beschriebenen Verfahren zur Relationsextraktion erfordern in der Trainingsphase große Textmengen, die mit möglichst vielen Beispielen der zu extrahierenden Relation annotiert sind. Daher wird analog zur Entwicklung bei der Extraktion von Eigennamen auch an **semi–überwachten Lernverfahren zur Relationsextraktion** geforscht, die nur eine Handvoll von Trainingsbeispielen benötigen, zusammen mit nicht-annotierten sehr großen Textmengen. Agichtein and Gravano [2000] haben schon recht frühzeitig ein Verfahren zur RE auf Basis von Bootstrapping präsentiert, das den seed-basierten Verfahren zur NER sehr ähnlich ist. Das Snowball genannte Verfahren startet mit einer kleinen Menge von Seed–Tupeln für eine zu extrahierende (binäre) Relation, z. B. korrekte Beispiele von Buch/Autor oder Firma/Hauptgeschäftsstelle. Snowball sucht dann nach Sätzen, in denen eines dieser Tupel auftritt,

[6] Hier kann nur stichwortartig auf diese aktuellen Entwicklungen eingegangen werden. Für ein vertiefendes Studium zum Thema "Maschinelles Lernen mit Kernels" vgl. man Schölkopf and Smola [2002] und Cristianini and Shawe-Taylor [2004].

und erzeugt entsprechende Muster, in denen die Tupelelemente durch ihre entsprechenden Typen ersetzt und die anderen Elemente des Satzes entsprechend generalisiert werden. Die so bestimmten Muster werden nun selbst auf das gesamte Korpus angewendet und führen eventuell zu einer Erweiterung der Seed–Liste. Dieser Prozess wird so lange wiederholt, bis keine neuen Seed–Elemente mehr gefunden werden. Ähnliche Verfahren zum Bootstrapping finden sich auch in Brin [1998], Riloff and Jones [1999], Yangarber et al. [2000] und Xu et al. [2007].

Allgemein funktionieren diese Verfahren umso besser, je redundanter und vielseitiger eine Relation im Korpus paraphrasiert wird. Es erscheint auch klar, dass je stereotypischer eine Relation ausgedrückt wird, desto weniger Seed–Elemente anfänglich benötigt werden. Ein großer Nachteil dieser Verfahren ist auch hier (wie bereits oben im Kontext von NER erläutert) die hohe Sensibilität der Verfahren bezogen auf die initiale Seed–Liste, insbesondere, wenn Relationen auf sehr unterschiedliche Weise in Texten ausgedrückt sind.

Ein alternatives semi–überwachtes Lernverfahren, das in Bunescu [2007] vorgestellt wird, ist bzgl. dieses Problems weniger sensibel. Dieses Verfahren basiert auf dem Maschinellen Lernansatz des "Multiple Instance Learning" (MIL, vgl. Dietterich et al. [1997]). Die einzige Überwachung für das MIL-Lernverfahren ist eine kleine Menge von Paaren von Eigennamen, für die bekannt ist, dass sie entweder zur extrahierenden Relation gehören (positive Beispiele) oder nicht (negative Beispiele). Beispielsweise zeigt die folgende Tabelle vier positive (mittels + gelabelt) und zwei negative (-) Beispiele:

+/-	Arg a_1	Arg a_2
+	Google	YouTube
+	Adobe Systems	Macromedia
+	Viacom	DreamWorks
+	Novartis	Eon Labs
-	Yahoo	Microsoft
-	Pfizer	Teva

Beispielpaare für positive und negative Firmenübernahmen.

Für jedes Paar wird nun eine Menge von Sätzen (auch Bag genannt) in einem Textkorpus bestimmt, in denen die zwei Argumente auftreten. Grundidee des MIL Verfahrens ist nun die Annahme, dass ein Textkorpus existiert, der ausreichend groß und divers genug ist, sodass angenommen werden kann, das in jedem Bag mindestens ein Satz enthalten sein wird, der explizit die angenomme Relation (oder ihre Negation) ausdrückt, ohne dass man das explizit manuell überprüfen muss. Ziel ist es nun, ausgehend von den Beispielpaaren und ihren assoziierten Bags automatisch ein RE–System zu induzieren, das akkurat für neue Sätze, in denen entsprechende Paare von Eigennamen genannt werden, entscheiden kann, ob die zugrundeliegende Relation gilt oder nicht. Bunescu [2007] verwendet nun MIL–Verfahren, in denen die zu lernende Entscheidungsfunktion mittels eines Kernels realisiert ist, ähnlich, wie er dies bereits für seine entsprechenden überwachten Lernverfahren zeigt, d. h. der Kernel betrachtet alle möglichen gemeinsamen Teilsequenzen zwischen zwei Sätzen. Auf diese Weise

können die charakteristischen Merkmale (in Form von Teilsequenzen, die auch Lücken enthalten dürfen) jeden Bags berechnet werden. "$\langle a_1 \rangle$ ... bought ... $\langle a_2 \rangle$ ... in ... billion ..." ist ein Beispiel für solch ein Merkmal. In seinen Experimenten benutzt Bunescu [2007] Sätze, die er mittels spezieller Suchanfragen an die Google–Suchmaschine bestimmt. Beispielsweise liefert die Suchanfrage "Google*******YouTube" Dokumente, in denen Sätze vorkommen, wo zwischen den Termen "Google" und "YouTube" maximal sieben beliebig andere Tokens vorkommen (* fungiert demnach als Platzhalter für ein einzelnes Token). Auf diese Weise bestimmt er pro Paar Bags, die aus mehreren hundert Sätzen bestehen können, wobei auffällig ist, dass die Bags für positive Beispiele in der Regel größer sind als für negative, siehe Bunescu [2007], Seite 119. Für zwei untersuchte Relationen (FirmenÜbernahme und PersonGeburtsort) erreicht er sehr gute Werte, die das Potential dieser Methode demonstrieren. Das Verfahren von Bunescu [2007] scheint sehr fehlertolerant zu sein. Daher können die Bags vollautomatisch bestimmt werden und sind zumindest für die angegebenen Beispiele überschaubar in ihrer Größe.

1.3.4 Extraktion komplexer Relationen

Die bisherigen Verfahren für RE erlernen in der Regel binäre Relationen. Es gibt bereits erste maschinelle Lernverfahren zur **Extraktion von komplexen Relationen** (auch bekannt als **Ereignisextraktion** oder **Szenario–Template–Filling**), vgl. u. a. McDonald et al. [2005], Melli et al. [2007] und Xu et al. [2007]. Als komplexe Relationen wollen wir beliebige mehrstellige Relationen betrachten, wobei einige der Argumente unspezifiziert sein können. Es ist klar, dass ein einzelner Satz nicht alle Argumente einer komplexen Relation ausdrücken muss, sondern es eher die Regel ist, dass die "Teile" einer komplexen Relation über mehrere Sätzen verteilt sind (wie in den Beispielen in Abschn. 1.1.3) oder aber dass die eine komplexe Relation ausdrückenden einzelnen Sätze zumindest sehr lang sind. Daher ist es nötig, potentielle Argumente aus unterschiedlichen Sätzen oder Teilsätzen zu vereinigen. In den älteren manuell erstellten IE–Systemen wurde dies meist mit einer einfachen Unifikationsstrategie behandelt: Wenn zwei Relationen identische Information in mindestens einem Argument haben und die übrigen Attribute konsistent sind, dann wird die gesamte Information beider Relationen mittels Unifikation vereinigt. Genauer wird für je zwei typkompatible Argumente überprüft, ob eine Koreferenzbeziehung besteht, ob sie semantisch kompatibel sind (via Domänenlexikon) oder ob sie in einem Subsumptionsverhältnis stehen. Darüber hinaus werden auch sehr anwendungsspezifische Heuristiken eingesetzt.

Ein elegantes Lernverfahren zur Extraktion von komplexen Relationen wird in McDonald et al. [2005] vorgestellt. Ausgangspunkt ist die Struktur der zu extrahierenden N–ären Relation in Form eines Tuples mit N Typen von Eigennamen, z. B. (Person, Job, Firma). Kernidee ist nun, ein N-äres Tupel in die Menge aller möglichen konsistenten binären Tupel zu faktorisieren, z. B. {(Person, Job), (Person, Firma), (Job, Firma)}, sodass in einem ersten Schritt alle möglichen entsprechenden binären Relationen in einem Text be-

stimmt werden können. Hierfür können im Prinzip die oben erwähnten binären Lernverfahren herangezogen werden. Jedes gefundene Paar kann dann als eine Kante eines Graphen betrachtet werden, wobei die Eigennamen den Knoten entsprechen. Der Algorithmus versucht dann die komplexen Relationen dadurch zu rekonstruieren, indem er die **maximalen Cliquen** im Graph berechnet, wobei Knoten mit gleicher Marke (also gleichem Eigenname) die Verbindungspunkte darstellen.[7] McDonald et al. [2005] betrachten auch gewichtete Cliquen, in denen die Kanten zwischen Knoten mit dem Wahrscheinlichkeitswert der zugehörigen binären Relation markiert ist. Dies erhöht die Fehlertoleranz, da nicht mehr eine perfekte binäre RE angenommen werden muss. Obwohl das Verfahren recht einfach ist, zeigt es eine vielversprechende Performanz, wie die Ergebnisse von Tests aus dem biomedizinischen Bereich zeigen (z. B. 0.6942 F–Maß für eine 4–stellige Relation). Ein weiterer Vorteil ist, dass das Verfahren nicht auf die Extraktion von komplexen Relationen aus einzelnen Sätzen beschränkt ist, wie dies etwa für die Verfahren von Melli et al. [2007] und Xu et al. [2007] gilt.

1.3.5 Domänenoffene Informationsextraktion

Die bisher betrachteten traditionellen Verfahren zur Informationsextraktion haben als Ausgangspunkt eine vordefinierte Relation und manuell überprüfte Beispiele in Form von Seeds oder annotierten Texten. Solche IE–Systeme sind damit sehr darauf spezialisiert, einzelne Relationen so effektiv wie möglich zu extrahieren. In der Regel müssen sie aufwändig angepasst werden, wenn sie zur Extraktion von neuen Relationen eingesetzt werden sollen. Auch operieren die meisten implementierten IE–Systeme noch auf recht kleinen Textkorpora von wenigen tausend Dokumenten. Da die IE–Technologie aber eine immer stärkere Rolle in Nachbargebieten, wie z. B. Text Mining, Antwortextraktion oder Ontologielernen einzunehmen beginnt, erhöht sich auch damit der Bedarf an IE–Technologien, die schnell und einfach für neue Aufgaben angepasst werden können und beliebig große Textmengen beherrschen. **Wichtige Anforderungen an die zukünftige IE–Technologie** sind dabei, dass sie 1) alle sinnvollen Relationen aus Texten extrahieren können (**domänenoffen**), 2) dabei keine relationsspezifische manuell erstellte Eingabe benötigen (**unüberwacht**) und 3) prinzipiell das gesamte Web betrachten können (**webbasiert**).

Beispielsweise präsentieren Shinyama and Sekine [2006] einen Ansatz zum **unbeschränkten Entdecken von Relationen**, der anstrebt, alle möglichen Relationen aus Texten zu extrahieren und diese in Tabellen zu präsentieren. Sekine [2006] hat diesen Ansatz weiterentwickelt, den er als **on-demand information extraction** bezeichnet. Ausgangspunkt für die Entwicklung dieses Ansatzes ist ein mittels Schlüsselwörtern spezifizierter Topik, der zum Aufsammeln relevanter Webseiten verwendet wird. Das System setzt dann dependenzbasiertes Parsing als zentrales Werkzeug ein, mit dem die syntaktische Struktur möglicher relevanter Relationen bestimmt wird. Die Kandidatenrelationen

[7] Cliquen sind Teilgraphen, in denen alle Knoten paarweise verbunden sind. Eine Clique C ist maximal, wenn es keine andere Clique gibt, die C enthält. Es gibt sehr effiziente Algorithmen zur Bestimmung aller maximalen Cliquen in einem Graphen.

werden dann mit spezialisierten Clustering-Algorithmen weiterverarbeitet. Ein ähnlicher Ansatz ist auch von Eichler et al. [2008] entwickelt worden, der diesen Ansatz mit fortgeschrittenen Benutzerinteraktionsmethoden verbindet.

Ein anderer Ansatz für das unüberwachte IE ist von Oren Etzioni und Kollegen entwickelt worden, vgl. Etzioni et al. [2005], Cafarella et al. [2005], Banko et al. [2007], Yates and Etzioni [2007] und Etzioni [2008]. Sie haben eine Reihe von Systemen entwickelt (z. B. KnowItAll, Textrunner, Resolver), welche als Ziel haben, große Mengen von Fakten (z. B. Namen von Politikern oder Wissenschaftlern) aus dem Web zu extrahieren, und zwar domänenunabhängig und hoch skalierbar. Ausgangspunkt ist eine kleine Menge von generischen Mustern (nur acht Muster der Art "X such as Y"). Diese Muster werden eingesetzt, um automatisch domänenspezifische Extraktionsmuster zu erlernen. Die zwei Kernkomponenten in dieser Systemfamilie sind der Extraktor und der Bewerter. Der Extraktor generiert aus den Schlüsselwörtern jeden Musters eine Suchanfrage an eine Suchmaschine, die zur Sammlung relevanter Dokumente benutzt wird. Es werden dann dieselben Muster zur Extraktion von Kandidatenfakten verwendet. Der Bewerter berechnet einen Wahrscheinlichkeitswert zur Bewertung der Korrektheit jeden Faktums bevor diese in die Wissensbasis integriert werden. Der Bewerter basiert seine Berechnung der Wahrscheinlichkeiten auf dem Suchmaschinentrefferzähler. Dieser wird zur Berechnung einer gegenseitig abhängigen Information zwischen extrahierten Instanzen einer Klasse und der Menge der automatisch bestimmten Diskriminationsphrasen herangezogen. Diese Art der automatischen Bewertung ist eine Erweiterung des Algorithmus PMI-IR, vgl. Turney [2001]. Mit Hilfe der gesamten Methode ist es möglich, Millionen von Fakten mit einem relativen Recall von ca. 80% und einer Präzision von ca. 90% zu bestimmen. Um die Performanz weiter zu erhöhen, werden spezifische Wrapper auf der Basis von Machinellen Lernverfahren eingesetzt, die Subklassen, Listen von Fakten und Definitionen extrahieren können. Ein ähnlicher Ansatz zur Extraktion von Definitionen und Listen aus Web-Snippets wird in Figueroa and Neumann [2008] vorgestellt (vgl. auch Abschn. 1.4.3).

Der zentrale Flaschenhals der von Etzioni und Kollegen entwickelten Methoden ist die Fokussierung auf unäre Relationen, obwohl sie behaupten, dass ihr Ansatz zumindest theoretisch für n-äre Relationsextraktion einsetzbar ist. In einem aktuellen Papier haben Rosenfeld and Feldman [2006] das System URES vorgestellt, dass ebenfalls zur Klasse der unüberwachten IE gehört. URES ist in der Lage, große Mengen von binären Relationen zu extrahieren, wie z. B. CEO_of, InventorOf, MayorOf, wobei sie Präzisionswerte in den oberen 80% erreichen. Weiterhin haben Downey et al. [2007] ein Verfahren vorgestellt, dass speziell zur Extraktion von spärlich vorkommenden Instanzen von Relationen eingesetzt werden kann. Sie zeigen insbesondere, wie unüberwachte Sprachmodelle zur Bewertung der Korrektheit verwendet werden können. Sie zeigen außerdem, dass mit Hilfe dieses neuen Ansatzes (den sie REALM nennen) eine Reduktion der Fehlerrate um 39% im Durchschnitt erreicht werden kann, wenn es mit dem Ergebnis des Systems Textrunner verglichen wird.

Das von Suchanek et al. [2007] entwickelte System YAGO ist ebenfalls ein interessanter Ansatz zur unüberwachten IE. Sie fokussieren auf die Extraktion

einer sehr großen Ontologie aus Wikipedia und Wordnet, wobei sie ähnliche Verfahren wie die oben erwähnten einsetzen. Sie beschreiben Experimente, in denen sie automatisch eine Ontologie mit mehr als 17 Millionen Entitäten und Relationen mit einer geschätzten Präzision von 95% extrahieren.

Allgemeiner betrachtet ist es nun weit akzeptiert, dass mit steigender Komplexität der zu extrahierenden Information auch die Komplexität der zugrunde liegenden linguistischen Merkmalsextraktion einhergehen muss, um jeweils optimalen Recall und Präzision zu erreichen. Die meisten der aktuellen Systeme zur Relationsextraktion setzen hierzu existierende Dependenzparser, wie z. B. MontyLingua, Stanford-Parser oder Minipar ein. Allerdings mangelt es diesen Verfahren meist am nötigen Grad an Abdeckung und Geschwindigkeit, um sie realistisch in großskalierten domänenoffenen IE–Systemen einzusetzen. Glücklicherweise weisen aktuelle Forschungsergebnisse in der Computerlinguistik auf neue Richtungen für das volle Parsing von großen Korpora hin, insbesondere im Bereich der Daten-gesteuerten Dependenzanalyse. Beispielsweise präsentiert Nivre [2008] einen inkrementellen Dependenzparser, der ein Sprachmodell zur Auswertung lokaler Determinismusentscheidungen heranzieht.

1.4 Domänenoffene Fragebeantwortung

1.4.1 Text-basierte Fragebeantwortung

Eine **textbasierte Fragebeantwortung** (**textQA**) empfängt frei formulierte, geschriebene Fragen in natürlicher Sprache (natural language, NL) und extrahiert die entsprechenden Antworten aus sehr großen Beständen von unstrukturierten Dokumenten.[8] Sie liefert präzise Antworten, also Textausschnitte, die genau der Antwort entsprechen (z. B. ein Personenname, ein Zeitausdruck oder eine definitorische Textpassage) und nicht nur Verweise auf Dokumente oder Paragraphen, die die Antworten (möglicherweise) enthalten, wie dies bei gängigen Suchmaschinen zur Zeit der Fall ist. TextQA–Systeme integrieren bereits eine Vielzahl von IR und IE Technologien. Ein typisches **domänenoffenes QA–System** umfasst dabei Kernkomponenten zur

1. **Frageanalyse**: Sie empfängt als Eingabe eine Frage Q in natürlicher Sprache und analysiert sie nach syntaktischen und semantischen Kriterien. Im Fall einer crosslingualen QA wird zusätzlich eine maschinelle Übersetzung der Frage durchgeführt (vgl. Abschn. 1.6). Als Ergebnis liefert die Frageanalyse ein Datenobjekt O_Q, das u.a. den Fragetyp (z. B. Faktenfrage oder Definitionsfrage), den erwarteten Antworttyp (mittels Eigennamenerkennung, z. B. Person oder Lokation), eine semantische Repräsentation (oft als spezifische Relationsextraktion, vgl. Abschn. 1.3.3), den Schlüsselwörtern und eventuell die Übersetzung beinhaltet.

2. **Dokumentensuche**: Die Schlüsselwörter werden in eine Anfrage eines

[8] Damit unterscheiden sie sich zentral von traditionellen QA–Systemen, die im Wesentlichen als Antwortquelle strukturierte Datenbanken heranziehen, vgl. auch Carstensen [2008] für einen historischen Überblick.

IR–Systems überführt (vgl. Absch. 1.2), das damit die relevanten Dokumente bestimmt. Hierbei können auch weitere Prozesse involviert sein, wie z. B. eine Expansion der Terme, umso eine größere Trefferquote zu erreichen.

3. Bestimmung **relevanter Textpassagen**: Es werden die Textpassagen bestimmt, die mit hoher Wahrscheinlichkeit die Antworten enthalten. Eine weit verbreitete Strategie ist es, die Textpassagen zu bestimmen, in denen möglichst viele Terme der Frage in unmittelbarer Nähe zu einander auftreten (sehr häufig wird hierzu auch der tf-idf–Wert herangezogen, vgl. Abschn. 1.2.1). Annahme ist hier, dass Textpassagen oder Sätze, die die Antwort enthalten oft eine hohe Ähnlichkeit mit der Frage haben ("Wie hoch ist das Matterhorn?", "Das Matterhorn ist exakt 4477,54 Meter hoch.").

4. **Extraktion von Antworten**: Antworten sind in der Regel Instanzen des erwarteten Antworttyps (EAT). Daher werden alle Instanznamen (die in der Regel Eigennamen entsprechen) normalisiert (siehe auch Abschn. 1.3.1, eventuell disambiguiert (vgl. Abschn. 1.3.2) und gewichtet. Die Gewichtung eines Kandidaten a_i wird in der Regel seine Häufigkeit umfassen, was der Annahme zugrunde liegt, dass eine hochfrequente Instanz eher die gesuchte Antwort repräsentiert als eine andere, die weniger häufig genannt wird. Darüber hinaus wird die Anzahl und die jeweilige Distanz zu den Fragetermen berücksichtigt, wobei auch syntaktische und semantische Ähnlichkeiten berechnet werden (z. B. mit dependenzbasierten Kernels, vgl. Abschn. 1.3.3).

5. **Präsentation der Antworten**: Auf Basis der Gewichte werden die Antwortkandidaten sortiert und die N–besten Kandidaten dem Benutzer präsentiert. Um das Vertrauen in die Richtigkeit einer Antwort a_i zu bekräftigen wird zusätzlich auch der Satz, in dem a_i auftritt als Kontext dargestellt. Dabei muss allerdings darauf geachtet werden, dass der Satz auch semantisch konsistent ist. Dies ist keine triviale Forderung, da für a_i ja sehr viele Sätze in Frage kommen (in Abhängigkeit der Häufigkeit) und die semantische Konsistenz variieren kann. In Abhängigkeit der Frage ist auch möglich, dass Antworten im Prinzip Zusammenfassungen von Texten entsprechen können und damit die Antwortpräsentation quasi einem Textzusammenfassungssystem entspricht, vgl. auch Abschn. 1.5.

Der aktuelle Erfolg in der Forschung von textQA–Systemen liegt zentral in der innovativen Kombination von Technologien aus verschiedenen Bereichen der Informatik, wie z. B. Information Retrieval, Informationsextraktion, Sprachtechnologie und Künstliche Intelligenz (cf. Moldovan et al. [2004]). Bedeutsam für den aktuellen Erfolg ist sicher auch die Tatsache, dass textQA–Systeme und Technologien seit 1999 im Rahmen der nordamerikanischen "Text REtrieval Conference" (TREC — http://trec.nist.gov/), sowie seit 2003 im Rahmen des europäischen "Cross–Language Evaluation Forum" (CLEF — http://www.clef-campaign.org/) systematisch evaluiert werden, vgl. auch Carstensen [2008].

1.4.2 Webbasierte QA

Die Adaption und Skalierung dieser neuen textQA–Technologien für das Web und damit für die Weiterentwicklung aktueller Suchmaschinentechnologien stellt allerdings enorme Herausforderungen bereit, insbesondere auch unter der Vorgabe, dass die Systemreaktionszeit nicht signifikant steigen darf (cf. Radev [2003]). Man betrachte beispielsweise die riesige Menge an Webinhalten, die die gegenwärtig besten Suchmaschinen zu indizieren in der Lage sind (Milliarden von Webseiten). Während sich eine automatische linguistische und semantische Indizierung von TREC ähnlichen Datenmengen (ca. 1GB Nachrichtentext) zur Beantwortung einer eingeschränkten Menge von Fragen mittels sprachtechnologischer Vorverarbeitung als sehr fruchtbar für die Entwicklung der textQA–Technologien erwiesen hat (wobei Aspekte der Systemreaktionszeit meist noch keine Rolle spielen), erscheint die gleiche Vorgehensweise für das Web zumindest gegenwärtig außer Reichweite. Zudem sind die TREC und CLEF Korpora statisch, da sie auf eine kurze und fixe Zeitspanne beschränkt sind. Im Web werden neue Dokumente permanent hinzugefügt und bereits existierende Dokumente können jederzeit verändert oder gelöscht werden. Systeme zur **webbasierten Fragebeantwortung** (**webQA**) sind damit verstärkt mit Problemen wie Redundanz, Aktualität und Zuverlässigkeit konfrontiert. Es muss dabei auch bedacht werden, dass ein Text in der Regel einen hohen Interpretationsspielraum umfasst, so dass eine statische, situationsunabhängige Analyse des Textes nur eingeschränkt möglich ist. Hinzu kommt die wachsende Mehrsprachigkeit der Webinhalte und der damit steigende Bedarf an einer crosslingualen Fragebeantwortung.

Es existieren bereits erste webQA–Systeme, die erfolgreich zeigen, wie durch Einsatz moderner Sprachtechnologie gepaart mit einer systematischen Ausnutzung der Redundanz der Webinhalte fortgeschrittene Technologien für zukünftige Antwortmaschinen erreichbar sind, vgl. Ankolekar et al. [2006], Clarke et al. [2001], Dumais et al. [2002], Kwok et al. [2001], Ravichandran and Hovy [2002], Neumann and Xu [2004], Wahlster [2007]. Diese Systeme verfügen über eine vergleichbare Architektur und führen im Wesentlichen die folgenden drei Schritte durch:

1. Transformation einer NL–Frage in eine Anfrage der spezifischen Suchmaschine.

2. Bestimmung von relevanten Webseiten durch eine standardisierte Suchmaschine, z. B. Google, Live Search, Yahoo.

3. Extraktion der Antworten aus den Webinhalten.

Der erste Schritt ist notwendig, um optimal die Anfragesyntax der jeweiligen Suchmaschine zu bedienen und um die Zugriffswahrscheinlichkeit auf relevante Dokumente zu erhöhen. Dieser letzte Aspekt kann auch als die Bestimmung einer "Antwortkontext–Voraussage" interpretiert werden. Beispielsweise wird hiermit eine NL–Frage *Wie heißt der deutsche Bundespräsident?* überführt in

der deutsche Bundespräsident heißt oder *Deutschlands Bundespräsident ist*. Ohne Berücksichtigung einer vorherigen Korpusanalyse entspricht diese Art der Transformation prinzipiell einer Versuch–und–Irrtum Strategie. Daher verwenden fortgeschrittene webQA–Systeme eine statistische Datenanalyse zur Bestimmung einer optimalen Gewichtung der einzelnen Antwortkontexte (cf. Dumais et al. [2002] und Ravichandran and Hovy [2002]).

Was den zweiten Schritt betrifft, so liefern Suchmaschinen in der Regel nur Verweise auf relevante Dokumente zusammen mit automatisch erstellten Textausschnitten (**Snippets** genannt). Sie bestehen aus kleinsten Textfragmenten, in denen Terme der Suchanfrage vorkommen. Daher werden in der Regel für den dritten Schritt zuerst die N–besten Dokumente auf den lokalen Rechner geladen, bevor die eigentliche Antwortextraktion stattfindet (cf. Kwok et al. [2001] und Ravichandran and Hovy [2002]). Die **Antwortextraktion** wird dann analog der textQA–Technologie mit Hilfe von kombinierten Sprach- und Wissenstechnologie–Komponenten durchgeführt.

Die Notwendigkeit, relevante Dokumente dynamisch zu laden, hat allerdings einen negativen Effekt auf die Reaktionszeit des gesamten webQA–Systems. Dumais et al. zeigen, dass für die Beantwortung von faktenbasierten Fragen die **Snippets als Antwortquellen** ausreichen können, wenn zum Einen eine Vielzahl von Antwortkontexten systematisch berechnet und zum Anderen eine große Menge von Snippets (einige hundert bis tausend) für die Antwortextraktion berücksichtigt wird (cf. Dumais et al. [2002]). Die Snippets enthalten Terme aus der Suchanfrage und heben diese entsprechend markiert hervor. Für die Antwortextraktion ist dies von Vorteil, da es oft der Fall ist, dass eine mögliche Antwort in unmittelbarer Nähe der Frageterme auftritt. Daraus kann aber nicht unmittelbar geschlossen werden, dass die im Kontext solcher Schlüsselwörter vorkommenden Terme automatisch sinnvolle Antwortkandidaten wären. Insbesondere kann daraus nicht abgeleitet werden, dass die Snippets der ranghöchsten Dokumente auch am wahrscheinlichsten die richtige Antwort enthalten.

Es ist ein zentrales Ziel der aktuellen Suchmaschinen, den besten Dokumenten einen hohen Rang zuzuordnen und nicht Dokumente zu präferieren, deren Snippets möglicherweise einen exakten Antwortstring enthalten (Suchmaschinen sind zur Zeit ja noch keine Antwortmaschinen). Daher kann die **Rangordnung**, wie sie durch aktuelle Suchmaschinen vorgegeben wird, nicht unmittelbar für die Bestimmung der Antwortkandidaten herangezogen werden. Denn es besteht ein relevanter Unterschied darin, ob die Suchmaschine aktuelle Webseiten oder Webseiten mit möglichst hoher Zugriffszahl präferiert. Dies ist z. B. für die Beantwortung von Fragen nach Personen des öffentlichen Interesses kritisch. Wurde beispielsweise gerade ein neuer Bundespräsident (oder Bundesligatrainer) gewählt, so ist die Zugriffszahl auf Webseiten mit diesem Inhalt zumindest vorübergehend niedrig im Vergleich zu Seiten, die auf vorherige Bundespräsidenten verweisen. Aus diesem Grunde kann man in der Antwortextraktion zwar die N–ersten Snippets betrachten, sollte danach aber deren Rangordnung ignorieren, vgl. Neumann [2008]. Das heißt das in diesem Fall die N–Snippets als Menge betrachtet wird. Es ist dann ein Ziel, während der Antwortextraktion eine **antwortspezifische Rangordnung** von relevanten Textausschnitten zu

bestimmen.

Aktuelle webQA–Systeme haben zur Zeit noch erhebliche Beschränkungen. Zum Einen betrifft das die Anfrage–gesteuerte Berechnung potentieller Antwortkontexte. Zum Anderen sind die Systeme in der Regel nur für die Verarbeitung bestimmter Arten von NL–Fragen (meist einfache faktenbasierte Fragen der Art "Wer/Wann/Wo") optimiert. Hinzu kommt, dass die aktuell relevanten Systeme monolinguale (überwiegend englischsprachige) Systeme sind.

1.4.3 Datengesteuertes QA

Um auch im Bereich der QA eine der IE vergleichbaren Leistungssteigerung bezüglich Adaptivität und Skalierbarkeit zu erreichen, stehen gegenwärtig datengesteuerte Methoden für end–to–end QA–Systeme im Fokus der Forschung und Entwicklung. Die eingesetzten Verfahren sind denen der Relationsextraktion und domänenoffenen IE sehr ähnlich, vgl. Abschn. 1.3.3 und 1.3.5.

Ramakrishnan et al. [2004] präsentieren einen Ansatz zum automatischen Erlernen von oberflächennahen Textmustern, die die Methode des Maximum Entropy Modellings zum Gewichten von Snippets heranzieht. Ihr QA–System erlernt automatisch Strategien, um Antwortpassagen zu entdecken und zu gewichten. Als Trainingsmaterial verwenden sie Frage/Antwortpaare von TREC–Korpora. Das sind in der Regel kurze Nachrichtentexte. Sie verwenden wissensintensive Methoden zur Merkmalsextraktion, u. a. volles Parsing und WordNet zur Disambiguierung.

Ein instanzbasierter Ansatz für QA ist von Lita and Carbonell [2004] entwickelt worden. Sie betrachten insbesondere temporalrestringierende Fragen aus den TREC–Korpora unter Verwendung eines sehr großen Fragekorpus. Sie stellen einen sehr interessanten Ansatz für das Clustern von Fragen vor, der auf dem strukturellen Vergleich von Fragen unter Verwendung von POS–Tagging und vollem Parsing basiert. Bei der Evaluation fokussieren sie auf webbasierte Snippets, die die korrekte Antwort enthalten, nicht jedoch auf die Extraktion der exakten Antworten.

In Figueroa et al. [2008] und Figueroa and Neumann [2008] werden Maschinelle Lernverfahren zur Beantwortung von Definitions- und Listenfragen aus Web–Snippets vorgestellt, die als gemeinsamen Kern eine Kombination von Clustering und Semantische Kernels realisieren. In einem ersten Schritt werden die Fragen automatisch so reformuliert, dass Standardsuchmaschinen optimal angesteuert werden können. Ähnlich dem Verfahren zur domänenoffenen IE (siehe Abschn. 1.3.5) wird eine sehr kleine Menge von generischen Reformulierungsmustern verwendet (z. B. "$\langle Q \rangle$ is a $\langle A \rangle$") mit denen konkrete Anfragen an eine Suchmaschine generiert werden. Die gleichen Muster werden ebenfalls zur Bestimmung der besten Textfragmente aus den gefundenen Snippets herangezogen, nachdem sie ebenfalls automatisch expandiert wurden. Zur Extraktion von Wörtern, die unterschiedliche Aspekte des Zielkonzeptes im Web repräsentieren, werden Korpus–basierte semantische Analysen (insbesondere Latent Semantic Analysis) und Disambiguierungsstrategien eingesetzt. Die Verfahren sind mit Frage/Antwort–Paaren der TREC und CLEF–Korpora evaluiert und die er-

zielten Ergebnisse sind vergleichbar mit denen der dort vorgestellten besten Systeme.

Basierend auf ihren eigenen Arbeiten an AskMSR erforschen Azari et al. [2003] die Verwendung von statistischen Modellen im Kontext von webbasierten QA–Systemen. Auch sie verfolgen eine strenge datengesteuerte Perspektive, die nur über sehr einfache sprachspezifische Wissensquellen verfügt. Im Wesentlichen führen sie einfache Reformulierungen und Mustervergleiche basierend auf N–gram Technologien durch. Damit generieren sie sehr elegant (sehr viele) Paraphrasen zur aktuellen Frage, die unmittelbar zur Suche mit einer IR–Suchmaschine verwendet werden. Um zu optimalen Reformulierungsstrategien zu gelangen, wird anschließend eine Kostenanalyse auf Basis von erlernten Entscheidungsbäumen für alle erfolgreichen Frage–Antwort Paare durchgeführt.

Echihabi and Marcu [2003] präsentieren eine Methode, die auf dem probabilistischen Noisy–Channel Model zur Bestimmung der Ähnlichkeit zwischen Fragen und Antwortsätzen basiert. Ihr bestes Ergebnis von MRR=0.812 wird für 24 Fragen von der TREC–11 auf Basis einer Trainingsmenge von ca. 100.000 Frage–Faktenantwort Paaren erzielt. Unter Verwendung eines Korpus von ca. 90.000 allgemeineren Frage–Antwort Paaren erreichen sie einen MRR zwischen 0.343 und 0.365 für 500 TREC–11 Fragen. Im Unterschied zu den oben erwähnten Verfahren verwenden sie sehr wissensintensive Verarbeitungskomponenten und eine sehr viel größere Trainingsmenge.

1.5 Textzusammenfassung

Die Ergebnispräsentation hat zum Ziel, die gefundenen Informationen (z. B. Dokumente, Textpassagen, Antworten, gefüllte Tabellen) dem Benutzer in verständlicher Form so darzustellen, dass möglichst rasch bewertet werden kann, ob die extrahierten Informationen tatsächlich relevant sind oder zumindest soweit hilfreich sind, dass der Benutzer sie für seine weitere Informationsrecherche nutzen kann. Daher ist es auch ein Ziel der Ergebnispräsentation, die gefundenen Informationen kompakt so **zusammenzufassen**, dass die wichtigsten Informationen "auf einen Blick" erkennbar sind. Dabei hängt es natürlich auch vom quantitativen Umfang der Information ab, wie einfach und vollständig dies geleistet werden kann.

Menschen produzieren beim Zusammenfassen meist kurze geschriebene oder gesprochene Texte, vgl. Endres-Niggemeyer [2004]. Wahrscheinlich auch deshalb, weil kurze Texte vom Menschen effektiv und schnell konsumiert werden können. Die **automatische Textzusammenfassung** beschäftigt sich mit Methoden der automatischen Erzeugung von texuellen Zusammenfassungen, wobei das Medium der Informationsquellen im Prinzip beliebig sein kann, z. B. Videos, Bilder, gesprochene oder geschriebene Sprache oder auch Kombinationen von verschiedenen Medien. In diesem Abschnitt wollen wir uns auf die Verarbeitung von geschriebener Sprache konzentrieren, also die automatische Erstellung von Textzusammenfassungen aus geschriebenen Textquellen.

Suchmaschinen versuchen dies aktuell z. B. dadurch zu leisten, indem neben der Rangordnung der Dokumente zu jedem Dokument bereits ein Textausschnitt

(**Snippet** genannt) berechnet wird, der die Terme der Anfrage im Kontext ihres Auftretens im Dokument zeigt, vgl. auch Abschn. 1.4.2 und Manning et al. [2008]. Die Snippets sollen dem Benutzer eine kurze und prägnante Erklärung liefern, warum das Dokument mit der Anfrage zusammenpasst. In diesem Sinne kann ein **Snippet** als kurze **Zusammenfassung** eines einzelnen Dokumentes betrachtet werden, bestehend aus dem Titel des Dokuments, dem Link zum Dokument und einem kleinen Textausschnitt. Im Prinzip kann man zwei verschiedene Methoden unterscheiden, wie diese kleinen Textausschnitte berechnet werden: **statisch** und **dynamisch**.

In einem statischen Verfahren wird — unabhängig von der konkreten Anfrage — stets ein bestimmer Ausschnitt aus den gefundenen Dokumenten bestimmt, z. B. bestehend aus den ersten N–Sätzen des Dokuments eventuell in Kombination mit vorhandener Meta–Information, wie etwa dem Typ des Dokumentes (PDF etc.) oder dessen Erzeugungsdatum. Eine statische Zusammenfassung wird oft zur Beantwortung von Definitionsfragen verwendet. Beispielsweise liefert die Anfrage "define:DFKI" an die Google–Suchmaschine als Ergebnis (Snippet hier kursiv hervorgehoben):

> Definitionen von **DFKI** im Web:
>
> - *Das Deutsche Forschungszentrum für Künstliche Intelligenz (DFKI) ist ein Public Private Partnership mit Großunternehmen, Mittelständlern, den ...*
> de.wikipedia.org/wiki/DFKI
>
> Definitionen finden zu **DFKI** in: Deutsch Englisch alle Sprachen

wobei dieser Text übereinstimmt mit der ersten Zeile in der entsprechenden Seite in Wikipedia.

Dynamische Verfahren berechnen die Snippets in Abhängigkeit einer konkreten Anfrage (engl. **query–biased summarization**), so dass sichergestellt wird, dass mindestens ein Term der Anfrage im Snippet enthalten ist. Dies ist quasi automatisch der Fall, da ja die Terme über den invertierten Index mit den Dokumenten direkt in Verbindung stehen, u. a. auch dadurch, dass zu jedem Term die Position im jeweiligen Dokument gespeichert ist (vgl. Abschn. 1.2). Damit kann sehr einfach und schnell zu jedem Term ein kleiner Textausschnitt bestimmt werden, z. B. in dem im Dokument die N–linken und rechten adjazenten Wörter eines Terms bestimmt werden. Für mehrere Terme, die im Dokument vorkommen, kann damit auch recht einfach bestimmt werden, ob sie in unmittelbarer Nähe zu einander stehen, z. B. im gleichen Satz. Wenn dies nicht der Fall ist und die jeweiligen Textausschnitte zudem nicht auf einander folgen, dann kann man dies mittels ... anzeigen, wie dies beispielsweise für folgenden Snippet gilt, der von der Google–Suchmaschine für die Anfrage "aktuelle Blue-ray Player" berechnet wird:

> Top 9: Die besten **Blu-ray-Player** im Test - CHIP Online

> 1. Dez. 2008 ... Wir haben die **aktuellen** Stand-Alone-Geräte unter die Lupe genommen und sagen, ... Wer über die Anschaffung eines **Blu-ray-Players** nachdenkt, ...

Zwar können mit diesen einfachen Methoden *sehr schnell* dynamische Snippets erzeugt werden, die auch für einen ersten Überblick sehr hilfreich sind. Allerdings können so zum Einen in der Regel keine kohärenten Texte erzeugt werden, und zum Anderen ist jeder Snippet mit einem einzelnen Treffer (einem einzelnen Dokument) direkt verknüpft. Damit werden sie erst dann für den Benutzer quasi sichtbar, wenn der entsprechende Ausschnitt der Trefferliste angezeigt wird.

Eine andere Möglichkeit der **Textzusammenfassung** wäre es daher, einen einzelnen, sprachlich und inhaltlich abgestimmten Text für eine Menge von Textdokumenten zu erzeugen. Die zentralen Probleme für solche **Zusammenfassungen aus mehreren Dokumenten** (engl. **multidocument summarization**) sind a) das Erkennen und die Behandlung von redundanten Textstellen, b) die Identifikation von wichtigen Unterschieden in den verschiedenen Dokumenten und c) das Sicherstellen von kohärenten Zusammenfassungen, sogar dann, wenn die Dokumente aus verschiedenen Quellen (Autor, Stilrichtung etc.) stammen, vgl. Radev et al. [2002].

Zur Bestimmung der Redundanz werden oft alle Paragrafen oder Sätze der Dokumente auf Grund ihrer Ähnlichkeit mittels Clustering gruppiert, vgl. auch Abschn. 1.2.4. Dann wird in einem nächsten Schritt für jedes relevante Cluster ein einzelner Text quasi als Stellvertreter bestimmt. In einem anschließenden Schritt werden dann die verschiedenen Textausschnitte zu einem Text zusammengefügt. Dies ist in der Regel ein sehr komplexer Prozess, da die Texte sprachlich und inhaltlich aufeinander abgestimmt werden müssen. Daher werden hierbei auch Verfahren der **Textgenerierung** eingesetzt, vgl. Abschn. **??** und Endres-Niggemeyer [2004] für weitere Details.

1.6 Multilinguales und sprachübergreifendes TIM

Multilingualität spielt im Zeitalter des Webs eine immer größere Rolle (vgl. dazu auch http://www.internetworldstats.com/stats.htm). Von wachsender Bedeutung sind dabei auch sprachübergreifende (**crosslinguale**) Verfahren mit Hilfe derer Informationsquellen aus verschiedenen Sprachen gemeinsam genutzt werden können. Kerntechnologien für ein TIM sind hierbei:

- **Crosslinguales Information–Retrieval** (CLIR): erlaubt Benutzern Anfragen in einer Sprache zu formulieren, die sie fließend beherrschen und führt eine Volltextsuche in Dokumenten durch, die in einer anderen Sprache formuliert sind.
- **Crosslinguales Question Answering** (CLQA): findet exakte Antworten in Dokumenten einer Sprache für natürlichsprachliche Fragen, die in einer anderen Sprache formuliert ist.

- **Crosslinguale Informationsextraktion** (CLIE): extraktiert und integriert relevante Informationen (Entitäten und Relationen) aus Dokumenten von verschiedenen Sprachen.

In allen Bereichen werden Verfahren der **Maschinellen Übersetzung** (MÜ) als zentrales Mittel zur Realisierung der Crosslingualität eingesetzt. In den Bereichen CLIR und CLQA werden dabei ähnliche Verfahren eingesetzt. Die meisten Verfahren gehen dabei von bilingualen Verfahren aus, d. h. die **Zielsprache** (die Sprache der Anfragen, z. B. Englisch) und die **Quellsprache** (die Sprache in denen die Dokumente verfasst sind, z. B. Deutsch) stammen jeweils aus einer einzigen aber jeweils unterschiedlichen Sprache. In CLQA wird häufig die Frage der Zielsprache zuerst mit MÜ–Verfahren in die Quellsprache der Dokumente übersetzt (also beispielsweise englische Fragen nach Deutsch), bevor der eigentliche Suchprozess gestartet wird, vgl. Sacaleanu and Neumann [2006].

Ein großes Problem für CLIR und CLQA ist es, dass die Anfragen in der Regel sehr klein sind (verglichen mit der Größe von Dokumenten, siehe auch Abschn. 1.2.1), sodass die Qualität der Übersetzung einen hohen Einfluss auf die Qualität des gesamten Systems hat. Beispielsweise kann ein Fehler in der Übersetzung zu weiteren Fehlern in der Frageexpansion und –umformulierung führen. Daher ist es auch eine häufig eingesetzte Strategie gerade in CLIR, die Dokumente der Quellsprache in die Zielsprache zu übersetzen und den Suchprozess in der Zielsprache durchzuführen, vgl. Hakkani-Tur et al. [2005]. Ein Vorteil ist dann auch, dass man die Ergebnisse aus verschiedenen Quellsprachen sehr einfach vermischen kann, da man ja nun nur einen einzigen Index zu berechnen und verwalten hat. Dabei kann man auch berücksichtigen, ob für Dokumente der Quellsprache bereits (manuelle) Übersetzungen vorliegen (so genannte parallele Texte). Damit kann man sich die Übersetzung ersparen und gleichzeitig den Redundanzwert (oder tf-idf–Wert, vgl. Abschn. 1.2.1) der Terme erhöhen.

Zentrales Ziel bei der CLIE ist die Identifikation und Extraktion von relevanter Information aus Dokumenten mit unterschiedlichen Ziel- und Quellsprachen. Sudo et al. [2003] zeigen, wie ein CLIE auf Basis eines CLIR realisiert werden kann. Ihr Ansatz ist quasi eine crosslinguale Version einer on–demand IE (ODIE, vgl. Abschn. 1.3.5). Die Kommunikation zwischen Benutzer und IE ist in englischer Sprache, d. h., die Anfrage in Form einer englischen Frage und die Ausgabe in Form von englischen Tabellen. Die eigentliche Extraktion findet in japanischen Texten mit Hilfe eines japanischen IE–Systems (JIE) statt. Daher wird zuerst die Anfrage ins Japanische übersetzt und damit die eigentliche IE angestoßen. JIE erzeugt Muster über Dependenzstrukturen dem Ansatz von ODIE folgend. Die erzeugten Tabelleneinträge werden dann Slot für Slot ins Englische übersetzt. Dieser CLIE–Ansatz kann auch als "anfragegesteuerte IE" (query–driven IE) oder **zielsprachenorientierter Ansatz** bezeichnet werden. Sudo et al. [2004] vergleichen diesen Ansatz mit **quellsprachenorientierten Ansätzen**, wo die Dokumente der Quellsprache in zielsprachliche Äquivalente übersetzt werden (also beispielsweise japanische Texte in äquivalente englische Texte). Ihre Experimente und Evaluationen zeigen, dass die anfragegesteuerte IE einen 8-10% höheren Recall aufweisen. Einer der Gründe ist hierbei die

Fehleranfälligkeit von MÜ–Systemen, insbesondere was die Übersetzung von Eigennamen betrifft. Große Probleme bewirkt ebenfalls die fehlerhafte Übersetzung von ganzen Sätzen, die relevante Relationen verbalisieren, da sie erhebliche Auswirkungen auf die nachfolgende Syntaxanalyse haben, sodass die Extraktion von Relationen in den übersetzten Dokumenten erschwert wird.

1.7 Perspektiven

Text–basiertes Informationsmanagement (TIM) integriert Technologien aus verschiedenen Bereichen der Sprachtechnologie, wie Information–Retrieval (IR), Informationsextraktion (IE), Fragebeantwortung (QA) und Textzusammenfassung (TZ). In diesem Artikel haben wir versucht zu zeigen, wie diese verschiedenen Perspektiven kohärent in einem Gesamtsystem zusammenspielen können. Dabei haben wir uns auf solche Aspekte konzentriert, die quasi als Basistechnologie betrachtet werden können. Viele wichtige Aspekte und Forschungsrichtungen konnten leider nur sehr knapp oder teilweise gar nicht angesprochen werden, wie z. B. Forschungs- und Entwicklungsrichtungen in den Bereichen des Semantic Webs, der Digitalen Büchereien oder spezifische Probleme und Lösungsvorschläge in bestimmten Bereichen, wie z. B. der Bioinformatik. Zusammenfassend für die zukünftige Forschung gilt, dass IR, IE, QA und TZ sich immer stärker aufeinander zu bewegen und verschmelzen werden.

1.8 Literaturhinweise

Die im Artikel angegebenen Literaturhinweise sollten bereits eine gute Basis für vertiefende Studien bieten. Darüberhinaus bieten die aktuellen Konferenzreihen TREC (Text Retrieval Conference, http://trec.nist.gov/), CLEF (Cross-Language Evaluation Forum, http://clef-campaign.org/) und TAC (Text Analysis Conference, http://www.nist.gov/tac/) einen sehr guten Einstiegspunkt in aktuelle Forschungsrichtungen und Entwicklungen. Da im Rahmen dieser Konferenzreihen auch regelmäßig vergleichende Systemevaluationen stattfinden, bieten sie auch einen sehr guten Einblick in die aktuelle Leistungsfähigkeit von IR, IE, QA und TZ Methoden. Weitere Hinweise zu Literatur und speziellen Webseiten kann man auch den Seiten von Wikipedia entnehmen, z. B. zum Thema “Information Retrieval”, “Information Extraction” (deutsche und englische Seiteneinträge) und “Question Answering” (nur englische Seite). Eine sehr informative Seite zum Thema “Text Summarization” ist über den Link http://www.summarization.com/ zu erreichen.

Literatur

Eugene Agichtein and Luis Gravano. *Snowball*: extracting relations from large plain-text collections. In *ACM DL*, pages 85–94, 2000.

A. Ankolekar, P. Buitelaar, P. Cimiano, P. Hitzler, M. Kiesel, M. Krötzsch, H. Lewen, G. Neumann, M. Sintek, T. Tserendorj, and R. Studer. Smartweb:

Mobile access to the semantic web. In *Proc. of the Demo Session at the International Semantic Web Conference*. Athens GA, USA, Nov., 2006.

D. Appelt and D. Israel. *Building information extraction systems*. Tutorial during the 5th ANLP, Washington, 1997. http://www.ai.sri.com/~appelt/ie-tutorial/.

Javier Artiles, Satoshi Sekine, and Julio Gonzalo. The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. In *the Workshop on Semantic Evaluation at ACL–2007, Prague, Chech*, 2007.

Masayuki Asahara and Yuji Matsumoto. Japanese named entity extraction with redundant morphological analysis. In *HLT-NAACL*, 2003.

D. Azari, E. Horvitz, S. Dumais, and E. Brill. A decision making perspective on web question answering. In *UAI*, 2003.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI*, pages 2670–2676, 2007.

Richard K. Belew. *Finding out about: a cognitive perspective on search engine technology and the WWW*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-63028-2.

Pavel Berkhin. A survey of clustering data mining techniques. In *acob Kogan, Charles Nicholas, and Marc Teboulle (eds.), Grouping Multidimensional Data: Recent Advances in Clustering*, pages 25–71, 2006.

Daniel M. Bikel, Richard M. Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231, 1999.

Andrew Borthwick, John Sterling, Eugene Agichtein, , and Ralph Grishman. Nyu: Description of the mene named entity system as used in muc-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

Sergey Brin. Extracting patterns and relations from the world wide web. In *WebDB*, pages 172–183, 1998.

Razvan C. Bunescu. *Learning fro Information Extraction: From Named Entity Recognition and Disambiguation To Relation Extraction*. PhD thesis, University of Texas at Austin, August 2007.

Razvan C. Bunescu and Raymond J. Mooney. Subsequence kernels for relation extraction. In *NIPS*, 2005.

Razvan C. Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. In *EACL*, 2006.

Razvan C. Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun K. Ramani, and Yuk Wah Wong. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–155, 2005.

Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. Knowitnow: Fast, scalable information extraction from the web. In *HLT/EMNLP*, 2005.

M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the AAAI–99*, pages 328–334, 1999.

Kai-Uwe Carstensen. *Sprachtechnologie – Ein Überblick*. Web publication, http://www.cl.uzh.ch/CL/carstens/Materialien/Sprachtechnologie.pdf, 2008.

Charles L. A. Clarke, Gordon V. Cormack, Thomas R. Lynam, C. M. Li, and G. L. McLearn. Web reinforced question answering (multitest experiments for trec 2001). In *TREC*, 2001.

M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999. URL `citeseer.nj.nec.com/collins99unsupervised.html`.

Nello Cristianini and John Shawe-Taylor. *Kernel Methods for Pattern Classification*. Cambridge University Press, Cambridge, MA, 2004.

S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP-CoNLL 2007*, pages 708–716, 2007.

Aron Culotta and Jeffrey S. Sorensen. Dependency tree kernels for relation extraction. In *ACL*, pages 423–429, 2004.

Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell.*, 89 (1-2):31–71, 1997.

Doug Downey, Stefan Schoenmackers, and Oren Etzioni. Sparse information extraction: Unsupervised language models to the rescue. In *ACL*, 2007.

Susan T. Dumais, Michele Banko, Eric Brill, Jimmy J. Lin, and Andrew Y. Ng. Web question answering: is more always better?. In *SIGIR*, pages 291–298, 2002.

Abdessamad Echihabi and Daniel Marcu. A noisy-channel approach to question answering. In *ACL*, 2003.

Kathrin Eichler, Holmer Hemsen, Markus Löckelt, Günter Neumann, and Norbert Reithinger. Interactive dynamic information extraction. In *KI*, pages 54–61, 2008.

B. Endres-Niggemeyer. *Automatisches Textzusammenfassen*, pages 410–432. Stauffenburg, Tbingen, 2004.

Oren Etzioni. Machine reading at web scale. In *WSDM*, 2008.

Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1):91–134, 2005.

Alejandro Figueroa and Günter Neumann. Finding distinct answers in web snippets. In *WEBIST (2)*, pages 26–33, 2008.

Alejandro Figueroa, Günter Neumann, and John Atkinson. Searching for definitions answers on the web using surface patterns. *IEEE Computer*, 2008.

Luciano Floridi. Is information meaningful data. *Philosophy and Phenomenological Research*, 70:351–370, 2005.

D. Freitag. Information extraction from html: Application of a general learning approach. In *Proceedings of the 15th AAAI*, 1998.

R. Grishman and B. Sundheim. Message Understanding Conference – 6: A Brief History. In *Proceedings of the 16th COLING*, Kopenhagen, Denmark, Europe, 1996.

Dilek Hakkani-Tur, Heng Ji, and Ralph Grishman. Information extraction to improve cross-lingual document retrieval. In *Proc. RANLP workshop on Multi-source, Multilingual Information Extraction and Summarization*, 2005.

S. Huffman. Learning information extraction patterns from examples. In Wermter, Riloff, and Scheler, editors, *Connectionist, Statistical, and Symbol Approaches to Learning for Natural Language Processing*, volume 1040 of *LNAI*, Berlin, Springer, 1996.

David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. Fastus: A cascaded finite-state transducer for extracting information from naturallanguage text. In *in Roche and Schabes (eds.) Finite State Devices for Natural Language Processing*, pages 383–406. MIT Press, 1996.

Cody C. T. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. In *WWW*, 2001.

Fan Li and Yiming Yang. A loss function analysis for classification methods in text categorization. In *ICML*, pages 472–479, 2003.

Wei Li and Andrew McCallum. Rapid development of hindi named entity recognition using conditional random fields and feature induction. *ACM Trans. Asian Lang. Inf. Process.*, 2(3):290–294, 2003.

L. Lita and J. Carbonell. Instance-based question answering: A data-driven approach. In *EMNLP 2004*, Barcelona, Spain, July 2004.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. URL `http://www-csli.stanford.edu/~hinrich/information-retrieval-book.html`.

Ryan T. McDonald, Fernando C. N. Pereira, Seth Kulick, R. Scott Winters, Yang Jin, and Peter S. White. Simple algorithms for complex relation extraction with applications to biomedical ie. In *ACL*, 2005.

Charles T. Meadow. *Text Information Retrieval Systems.* San Diego: Academic Press, 1992.

Gabor Melli, Martin Ester, and Anoop Sarkar. Recognition of multi-sentence n-ary subcellular localization mentions in biomedical abstracts. In *LBM (Short Papers)*, 2007.

D. Moldovan, S. Harabagui, C. Clark, M. Bowden, J. Lehmann, and J. Williams. Experiments and analysis of lcc's two qa systems over trec 2004. In *TREC 2004*, Gaithersburg, USA, 2004.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224, 2008.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January 2007. ISSN 0378-4169.

G. Neumann, R. Backofen, J. Baur, M. Becker, and C. Braun. An information extraction core system for real world german text processing. In *Proceedings of 5th ANLP*, Washington, USA, March 1997.

Günter Neumann. *A Hybrid Machine Learning Approach for Information Extraction from Free Texts.*, pages 390–397. Studies in Classification, Data Analysis, and Knowledge Organization. Springer-Verlag Berlin, Heidelberg, New-York, 2006.

Günter Neumann. Strategien zur webbasierten multilingualen fragebeantwortung. *Computer Science - Research and Development*, 22(2):71–84, 2008.

Günter Neumann and Feiyu Xu. Mining natural language answers from the web. *International Journal of Web Intelligence and Agent Systems*, 2004.

Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553, 2008.

D. Radev. Panel on web-based question answering. In *AAAI Spring Symposium on New Directions in Question Answering*, 2003.

Dragomir R. Radev, Eduard Hovy, and Kathleen McKeown. Introduction to the special issue on summarization. *Comput. Linguist.*, 28(4):399–408, 2002.

G. Ramakrishnan, D. Paranjape, S. Chakrabarti, and P. Bhattacharyya. Is question answering an acquired skill? In *WWW04*, 2004.

Deepak Ravichandran and Eduard H. Hovy. Learning surface text patterns for a question answering system. In *ACL*, pages 41–47, 2002.

Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, pages 474–479, 1999.

Binyamin Rosenfeld and Ronen Feldman. Ures : an unsupervised web relation extraction system. In *ACL*, 2006.

Bogdan Sacaleanu and Günter Neumann. A cross-lingual german-english framework for open-domain question answering. In *CLEF*, pages 328–338, 2006.

Bernhard Schölkopf and Alex Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

Satoshi Sekine. On-demand information extraction. In *ACL*, 2006.

Satoshi Sekine, Kiyoshi Sudo, and Chikashi Nobata. Extended named entity hierarchy. In *The Third International Conference on Language Resources and Evaluation*. Canary Island, Spain, 2002.

Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *HLT-NAACL*, 2006.

S. Soderland. *Learning Text Analysis Rules for Domain Specific Natural Language Processing*. PhD thesis, University of Massachusetts Amherst, 1997.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. pre-codie–crosslingual on-demand information extraction. In *HLT-NAACL*, 2003.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. Cross-lingual information extraction system evaluation. In *Proceedings of Coling 2004*, pages 882–888, Geneva, Switzerland, Aug 23-Aug 27 2004. COLING.

Peter D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *ECML*, pages 491–502, 2001.

Wolfgang Wahlster. Smartweb: multimodal web services on the road. In *ACM Multimedia*, page 16, 2007.

Rui Wang and Günter Neumann. Recognizing textual entailment using a subsequence kernel method. In *AAAI*, 2007.

Feiyu Xu, Hans Uszkoreit, and Hong Li. A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In *ACL*, 2007.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. Unsupervised discovery of scenario-level patterns for information extraction. In *Proceedings of the 6th ANLP*, Seattle, USA, April 2000.

Roman Yangarber, Winston Lin, and Ralph Grishman. Unsupervised learning of generalized names. In *COLING*, 2002.

Alexander Yates and Oren Etzioni. Unsupervised resolution of objects and relations on the web. In *HLT-NAACL*, 2007.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, 2003.

Shubin Zhao and Ralph Grishman. Extracting relations with integrated information using kernel methods. In *ACL*, 2005.