Übungsblatt 4

Vorlesung Intelligente Systeme im WWW - SS 2001

12. Juni 2001

Modalitäten: Abgabe des Lösungsblattes bis zum 20.6.2001 - 15:45 per E-Mail an volz@aifb.uni-karlsruhe.de oder schriftlich zu Beginn der Übung.

1 Aufgabe 1

Modalitäten Richtig beantwortete Frage: 3 Pt., Falsch beantwortete Frage: entsprechender Punktabzug , keine Antwort: 0 Pt. Ein negatives Gesamtergebnis wird nicht übertragen.

1.1 Kontext

In der Vorlesung wurden zwei Maße aus dem Information Retrieval zur Bewertung der Häufigkeit von Termen vorgestellt:

1. Termhäufigkeit (TF)
2. Termhäufigkeit / Inverse Dokumenthäufigkeit (TFIDF)

Ein Beispiel für Information Retrieval Systeme sind Internet-Suchmaschinen wie www.google.com oder www.altavista.com. Dokumente D werden von solchen Retrieval-Systemen als Termvektoren dargestellt:

$$D = (t_i, t_j, ..., t_p)$$

Hierbei ist jedes t_k ein in einem Dokument D auftretender Term.

In gleicher Art und Weise werden auch Benutzeranfragen (Queries) Q in Vektorform dargestellt:

$$Q = (q_a, q_b, ..., q_r)$$

Auch hier ist jedes q_k ein Term in einer Anfrage Q.

Wenn die Gesamtheit der Terme im System bekannt ist, lassen sich obige Vektoren oftmals auch wie folgt darstellen:

$$D = (t_0, w_{d_0}; t_1, w_{d_1}; ...; t_t, w_{d_n})$$

$$Q = (q_0, w_{q_0}; q_1, w_{q_1}; ...; q_t, w_{q_n})$$

Dabei beichnet w_{d_k} das Gewicht von Term t_k in Dokument D (respektive Anfrage Q). Es wird vorausgesetzt, dass w_{d_k} den Wert 0 annimmt, wenn der Term t_k in einem Dokument (oder einer Anfrage) nicht auftritt (1 sonst).

Wenn man letztere Vektordarstellungen benutzt, lässt sich die Übereinstimmung zwischen einem bestimmten Dokument D und einer gegebenen Anfrage Q leicht durch folgende Formel berechnen:

$$similarity(Q, D) = \sum_{i=1}^{n} w_{q_i} * w_{d_i}$$

Falls die Termgewichte also nur die Werte 0 und 1 annehmen können, gibt die Formel an, wieviele Terme sowohl in D als auch in Q vorkommen.

In der Praxis ist die binäre Bewertung von Termen (also $w_{d_k} \in \{0, 1\}$) nicht ausreichend. Beispielsweise sollen dem Benutzer die relevantesten Dokumente an erster Stelle präsentiert werden. Normalerweise werden dazu die Terme bezüglich ihrer Wichtigkeit bewertet, um die Retrieval-Effektivität zu erhöhen. Die Effektivität wird meist an zwei Kriterien gemessen:

1. Nur relevante Terme sollen gefunden werden.

2. Nicht-relevante Terme sollen nicht gefunden werden.

Um die Ausgabe eines Systems bezüglich dieser Kriterien zu bewerten, werden meist zwei Maße herangezogen: *Precision* and *Recall*.

Precision misst den Anteil der relevanten Dokumente an der Gesamtheit der ausgegebenen Dokumente.

Recall misst den Anteil der relevanten Dokumente in der Ausgabe an der Gesamtheit aller relevanten Dokumente im System.

Im Prinzip soll ein System sowohl einen hohen *Recall* produzieren (alles Relevante ausgeben, wichtigstes zuerst), als auch einen hohe *Precision* erreichen (nichts Falsches ausgeben). Dies lässt sich in der Praxis meist nicht erreichen, vielmehr muss man eine Balance zwischen Precision and Recall finden, deshalb werden Termgewichtungsverfahren wie Term Frequency (TF), Term Frequency Inverse Document Frequency (TFIDF) u.a. benutzt.

Term Frequency (TF) gewichtet Terme mit der Häufigkeit ihres Auftretens in einem Dokument.

Term Frequency Inverse Document Frequency (TFIDF) gewichtet TF mit einem zusätzlichen Faktor (IDF), der bewirkt, dass Terme, die nur in wenigen Dokumenten auftreten, gegenüber Termen, die in mehreren Dokumenten auftreten, "bevorzugt" werden.

Zur Beantwortung der Fragen setzen Sie bitte die meistgebrauchte Formel für die Berechnung der IDF $log(N/n)$ voraus, dabei ist N die Gesamtzahl der Dokumente in einer Dokumentenmenge und n die Zahl der Dokumente in der Dokumentenmenge, in denen der betrachtete Term vorkommt.

1.2 Frage 1

Die Verwendung von TF erhöht den Recall.

1.3 Frage 2

Geben Sie eine Situation an, in der TF keine akzeptable Precision erzielt.

1.4 Frage 3

Geben Sie eine Situation an, in der eine Normalisierung von Termgewichten bezüglich der Länge des Dokumentenvektors sinnvoll ist.

1.5 Frage 4

Welches Termgewichtungsverfahren ist in Situationen zu bevorzugen, in der sich der Dokumenteninhalt oft ändert ?

1.6 Frage 5

Eine Normalisierung von Termgewichten bezüglich der Länge des Anfragevektors ist sinnvoll.

2 Aufgabe 2 (24 Pt.)

2.1 Kontext

Das in Ontologien-gespeicherte Wissen wird typischerweise durch Inferenzregeln abgefragt. Durch die Kombination von Ontologien mit Inferenzregeln lassen sich mit deklarativen Mitteln intelligente Internet-Agenten erstellen.

In F-Logic definierte Inferenzregeln für RDF(S) können beispielsweise von der Java-Inferenzmaschine SiLRI [1] interpretiert werden.

SiLRI arbeitet wie folgt: Zunächst werden RDF(S)-Dateien in den Hauptspeicher eingelesen, dabei wird jedes RDF Statement in Fakten der Form `setatt_(Resource, Property, Value)` übersetzt. Auf diesen Statements können sie F-LOGIC-Prädikate definieren, die beispielsweise für Anfragen an die Ontologie benutzt werden können.

Als Beispiel werde die RDFS-SubClassOf-Property an ein Prädikat gebunden (unter Verwendung einer Hilfsregel `s(X,Y,Z)<-setatt_(X,Y,resource(Z)).`):

```
subClassOf(X,Z)<-s(X,http://www.w3.org/2000/01/rdf-schema#subClassOf,Z).
```

Da die subClassOf-Property transitiv ist, benötigt man noch:

```
subClassOf(X,Z) <- subClassOf(X,Y) & subClassOf(Y,Z).
```

Anmerkung: Die Hilfsregel dient ausschließlich der Vereinfachung und ist nicht zwingend notwendig, hilft Ihnen aber eventuell in der Lösung der Aufgabe. Beachten Sie, dass Variablen als Großbuchstaben geschrieben werden und die Regeln mit einem Punkt abgeschlossen werden.

2.2 Aufgabe 2.1 (4 Pt.)

Definieren Sie ein InstanceOf-Prädikat für die RDF-Schema-Property

```
http://www.w3.org/2000/01/rdf-schema#instanceOf
```

2.3 Aufgabe 2.2 (12 Pt.)

Definieren Sie Prädikate für RDF-Properties und RDFS-Subproperties.

Anmerkung: Sie könnten propertyOf als dreistelliges Prädikat `propertyOf(X,Y,Z)` definieren, wobei die Variablen folgende Aufgabe hätten: X bindet den Property-Namen, Y die Domäne und Z den Range der Property.

[1] http://www.ontoprise.de/download/SiLRI.zip inklusive F-Logic Tutorial

2.4 Aufgabe 2.3 (8 Pt.)

Berücksichtigen Sie nun die in RDF-Schema definierte Auto-Ontologie aus der beiliegenden Datei. Sie können mittels der oben erstellten Regeln Anfragen an die Ontologie stellen. Stellen Sie folgende Anfragen:

1. Ermittle alle Eigenschaften und deren Werte der `A_KLASSE`.
2. Ermittle die Klasse welcher der `VOLVO_V70` angehört.

Beispiel: Die folgende Anfrage gibt die Farbe des `VOLVO_V70` aus:

```
<- s(file:auto.rdf#VOLVO_V70, file:auto.rdf#HAT_FARBE, F).
```

3 Aufgabe 3 (21 Pt.)

3.1 Kontext

Das folgende Schaubild zeigt noch einmal die verschiedenen Bestandteile von Ontologie-basierten Systemen:

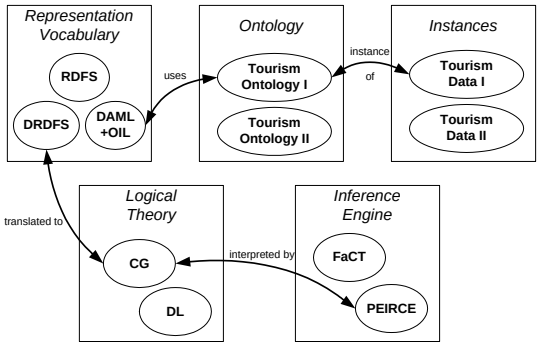

Abbildung 1: Repräsentationsvokabulare, Ontologien und Instanzen

1. Instanzen instanziieren in der Ontologie definierte Begriffe (analog: Objekte instanziieren Klassen in Objekt-orientierten Systemen).
2. Eine Ontologie definiert Begriffe und deren Beziehungen, dazu wird ein bestimmtes Repräsentationsvokabular benutzt (in Aufgabe 2 war das RDF(S)).

3. Ein Repräsentationsvokabular hat eine bestimmte Semantik, diese Semantik wird für Maschinen verständlich, wenn Sie in Logiksprachen wie Terminologische Logiken (DL), Begriffliche Graphen (CG), Frame-Logic (siehe Aufgabe 2) formal definiert wird.

4. Bestimmte Logiksprachen werden von Inferenzmaschinen (wie SiLRi oder OntoBroker (F-Logic), FaCT (DL) oder Peirce (CG)) interpretiert.

Entsprechend braucht man für die Umsetzung eines Ontologie-basierten Systems für jede genannte Komponente einen Vertreter und muss die Abbildung zwischen den Komponenten leisten. Die Kanten zwischen Inferenzmaschine, Logiksprache und Repräsentationsvokabular erledigen sich (in der Zukunft) wohl von selbst. In Aufgabe 2 haben Sie (hoffentlich) gerade RDF-Schema an F-Logic angebunden, welche von der Inferenzmaschine SiLRI interpretiert werden kann. Die Kanten zwischen Instanzen, Ontologien und Repräsentationsvokabularen ergeben sich durch die Referenzierung von Namespaces in RDF Dateien, alternativ kann man natürlich auch alles in eine Datei schreiben (siehe das Auto-Beispiel in Aufgabe 2, welches nicht zwischen Instanzen und Ontologie trennt), was man aber aus Gründen der Wartbarkeit tunlichst sein lassen sollte.

3.2 Die Aufgabe

Dieses Mal sollen Sie für eine gegebene Ontologie-Struktur mit algebraischer Semantik ein geeignetes Repräsentationsvokabular finden. Wegen Ihres umfangreichen Vorwissens kommt Ihnen bei Betrachtung der folgenden Definition natürlich sofort das RDF-Schema-Vokabular in den Sinn.

Identifizieren Sie für alle Elemente der gegebenen Struktur die entsprechenden Elemente des RDF-Schema-Vokabulars.

Warum ist die Eigenschaft der Disjunktheit zwischen den Mengen C und R bei Verwendung von RDF automatisch gegeben ?

3.3 Definition

Eine *Ontologie* habe folgende Struktur $\mathcal{O} := (C, \leq_C, R, \sigma, \leq_R)$, wobei

- die Menge C als die Menge der *Begriffsidentifikatoren* bezeichnet werde

- die Menge R als die Menge der *Relationsidentifikatoren* bezeichnet werde,
- die Mengen C und R disjunkt seien,
- die partielle Ordnung $\leq_C$ auf C als *Begriffshierarchie* oder *Taxonomie* bezeichnet werde,
- eine Funktion $\sigma\colon R \to C \times C$ *Signatur* genannt werde,
- eine partielle Ordnung $\leq_R$ auf R als *Relationenhierarchie* bezeichnet werde, hierbei impliziere $r_1 \leq r_2$, daß $\sigma(r_1) \leq_{C\times C} \sigma(r_2)$ für alle $r_1, r_2 \in R$ gelte.

Die Funktion $\mathrm{dom}\colon R \to C$ mit $\mathrm{dom}(r) := \pi_1(\sigma(r))$ gibt den *Ursprung* von r zurück, die Funktion $\mathrm{range}\colon R \to C$ mit $\mathrm{range}(r) := \pi_2(\sigma(r))$ gibt das *Bild* von r zurück.