Exposé: Performanzvergleich verschiedener Ansätze der Suche in annotierten Korpora

Matthias Bindernagel

1. September 2006

1 Hintergrund

Das Projekt *Deutsch Diachron Digital* [4] hat es sich zur Aufgabe gemacht ein umfangreiches, diachrones Korpus der deutschen Sprache zu erstellen.

Mit Hilfe dieses Korpus, welches sich dem Benutzer in Form einer Webanwendung präsentiert, sollen linguistische Frage- und Problemstellungen bearbeitet werden können. Um diesen Zweck zu erfüllen enthält das Korpus Dokumente in verschiedenen Fassungen, die durch strukturelle, linguistische und graphische Annotationen auf verschiedenen Ebenen weiter attributiert werden, wie in [1] vorgestellt.

Anfragen gegen das Korpus beziehen sich somit zum einen auf textuelle Informationen sowie auf damit assoziierte Attribute in Form von Annotationen. Das bedeutet, dass Texte und in ihnen enthaltene Textabschnitte durch folgende Suchmechanismen selektiert werden können: Für das Matching von reinem Text wird eine Volltextsuche nach exakten Zeichenketten, sowie nach flexibleren Formen, wie regulären Ausdrücken oder Wildcards benötigt. Weiterhin sollen Texte beziehungsweise Textabschnitte anhand ihrer linguistischen und strukturellen Eigenschaften, also anhand der mit ihnen assoziierten Annotationen, ausgewählt werden. Zuletzt sollen beide Suchoptionen kombiniert werden können, um eine Auswahl von Textabschnitten anhand all ihrer vorhandenen Eigenschaften zu ermöglichen.

2 Ziel

Anliegen der geplanten Arbeit ist es, die Performanz[1] und Praktikabilität zweier Implementationen der Suche auf einer annotierten Textsammlung zu evaluieren. Dabei muss der Einsatz des Korpus in Form einer Webanwendung Beachtung finden: Antwortzeiten sollten vertretbare Werte einhalten, auch eine hohe Anzahl parallel stattfindender Anfragen sollte den Service nicht unbenutzbar machen.

Für die Speicherung und Verwaltung der Annotationsdaten bieten sich relationale Schemata und XML-basierte Ansätze an. Für eine effektive Volltextsuche werden Implementationen benutzt, die auf inversen Indizes beruhen. Diese verschiedenen Paradigmen beziehungsweise Implementationen sollen im Rahmen dieser Arbeit gegenübergestellt werden und es soll damit einhergehend eine Bewertung der Vor- und Nachteile dieser Möglichkeiten stattfinden.

[1] in Hinsicht auf Antwortzeit, Mehrbenutzerfähigkeit und Ressourcenverbrauch

3 Vorgehen

3.1 Das Referenzkorpus

Um die Anforderungen an die Suchmechanismen bereits in der Entwurfsphase des eigentlichen Korpus bewerten zu können, werden Benchmarks anhand eines Referenzkorpus angestellt. Hierfür wird das *Tübinger Partiell Geparstes Korpus des Deutschen/Schriftsprache* [8], kurz *TüPP-D/Z*, benutzt. Dieses Korpus ist eine Sammlung von Texten aus der Tageszeitung TAZ und wurde hinsichtlich Satzstruktur und topologischen Feldern annotiert. Es umfasst alle Artikel in einem Zeitraumes von über 12 Jahren und beinhaltet mehr als 200 Millionen Wortvorkommen.

3.2 Datenhaltung und -verwaltung

Für die Bereithaltung der Text- und Annotationsdaten werden, wie bereits erwähnt, zwei verschiedene Möglichkeiten wahrgenommen.

Bei Verwendung des relationalen Ansatzes werden die im Korpus enthaltenen Texte und Annotationen in der Datenbank *Oracle Database 10g* [7] abgelegt. Das für die Speicherung benutzte Datenbankschema wird sich an den für *Deutsch Diachron Digital* bereits entwickelten Konzepten orientieren, wie in [2] und [3].

Für die Datenhaltung im XML-Format wird ein an das Format des *TüPP-D/Z*-Korpus angelehntes Format im Rahmen dieser Arbeit entwickelt werden.

3.3 Volltextsuche

Für die auf inversen Indizes basierende Volltextsuche stehen verschiedene Lösungen zur Verfügung: *Oracle Database 10g* stellt mit *Oracle Text* [5] bereits integrierte, umfangreiche Textsuchoptionen zur Verfügung.

Ein weiteres Produkt, welches sich für den Einsatz im Projekt *Deutsch Diachron Digital* anbietet, ist die Java-Volltextsuchmaschine *Apache Lucene* [6]. Es zeichnet sich durch seine hohe Performanz, Verbreitung und die Möglichkeit der Verwendung regulärer Ausdrücke[2] aus und wird somit als zweiter Kandidat in Verbindung mit dem XML-basierenden Korpus getestet werden.

3.4 Umgebung

Um einen angemessenen Vergleich beider vorgestellter Suchmechanismen zu ermöglichen wird eine Testumgebung basierend auf drei Schichten aufgebaut.

Die untere Schicht umfasst das Korpus in seiner jeweiligen Implementation und die damit verbundenen Suchmaschinen *Oracle Text* und *Apache Lucene*. In der mittleren Schicht werden die Anfragen an das Korpus verarbeitet. Dazu stellt diese eine Schnittstelle für die obere Schicht zur Verfügung mit der Anfragen formuliert und ausgeführt werden können. Sie setzt diese Anfragen dazu in spezifischere Anfragen auf Basis von *Oracle*/*Oracle Text* respektive XML/*Apache Lucene* um und abstrahiert damit die verschiedenen Implementationen zugunsten einer unabhängigen Schnittstelle. Um exemplarische Anfragen an das Korpus auszuführen wird die obere Schicht benutzt. Innerhalb dieser wird

[2] ab *Apache Lucene* Version 1.9

die unabhängige Schnittstelle der mittleren Schicht benutzt. Die Aufrufe der Methoden dieser Schnittstelle werden als Anhaltspunkte für Performanzmessungen dienen, d.h. die Durchführung der Messungen und deren Auswertung werden in dieser beziehungsweise mit Hilfe dieser Schicht erfolgen. Um vergleichbare Aussagen über die Performanz zu treffen werden die Antwortzeiten für verschiedene Anfragen gemessen. Dies wird unter Verwendung eines Messframeworks, wie *Apache JMeter* [9], geschehen.

3.5 Repräsentative Anfragen

Um aussagekräftige Messergebnisse über die Performanz der zwei möglichen Suchmechanismen zu erhalten, werden verschiedene Arten von repräsentativen Anfragen gestellt. Zum Einen werden reine Textsuchen durchgeführt, d.h. exaktes Stringmatching sowie das Matching von regulären Ausdrücken[3]. Des Weiteren werden Anfragen gestellt werden, die auch auf die Kombination von Textsuche und Matching von Annotationen beziehen, d.h. die Integration der Textsuche in die Filterkriterien einer SQL- beziehungsweise XQuery-Anfrage untersuchen. Dafür werden verschiedene Annotationstiefen angefragt, um eventuelle Performanzeinbrüche in den verschiedenen Umsetzungen der Beantwortung von Annotationsanfragen auszumachen, d. h. an dieser Stelle wird ein direkter Vergleich des relationalen Systems mit dem XML-basierten Ansatz der Annotationsspeicherung und -anfrage durchgeführt. Die Verwendung eines Query-Generators[4] erscheint hier sinnvoll.

3.6 Benchmark und Auswertung

Beide Varianten werden jeweils bezüglich verschiedener Kriterien untersucht: Antwortzeit, Mehrbenutzerfähigkeit[5], Ressourcenverbrauch und Praktikabilität der Implementation. Die genaue Art der Auswertung der genannten Kriterien wird im Verlauf der Arbeit weiter verfeinert werden, da genauer zu untersuchende Engpässe während der Umsetzung auftauchen können.

[3] solang es der Suchmechanismus unterstützt, alternativ steht die Verwendung von Wildcards in Suchausdrücken zur Verfügung

[4] zur Generierung von Anfragen aus Templates

[5] Antwortzeit bei massiv parallelen Anfragen

Literatur

[1] Lüdeling, Anke, Poschenrieder, Thorwald und Faulstich, Lukas (2004): *DeutschDiachronDigital - Ein diachrones Korpus des Deutschen*. Jahrbuch für Computerphilologie.
`http://www.deutschdiachrondigital.de/publikationen/ddd-computerphilologie.pdf`

[2] Vitt, Thorsten (2005): *DDDquery: Anfragen an komplexe Korpora*. Diplomarbeit, Humboldt-Universität zu Berlin, Institut für Informatik, Berlin.
`http://www.deutschdiachrondigital.de/publikationen/dddq.pdf`

[3] Faulstich, Lukas, Leser, Ulf und Vitt, Thorsten (2006): *Implementing a Linguistic Query Language for Historic Texts*. Query Languages and Query Processing (QLQP-2006): 11th Intl. Workshop on Foundations of Models and Languages for Data and Objects (FMLDO).
`http://www.deutschdiachrondigital.de/publikationen/qlqp.pdf`

[4] `http://www.deutschdiachrondigital.de/`

[5] `http://www.oracle.com/technology/products/text/index.html`

[6] `http://lucene.apache.org/`

[7] `http://www.oracle.com/database/index.html`

[8] `http://www.sfs.uni-tuebingen.de/de_tuepp.shtml`

[9] `http://jakarta.apache.org/jmeter/`