Erstellen eines Linked-Data-Fußballdatensatzes

Seminararbeit im Seminar

Semantic Media Mining

Wintersemester 2012/13

Hasso-Plattner-Institut Potsdam

vorgelegt von

Tanja Bergmann

Stefan Bunk

Johannes Eschrig

Daniel Roeder

Ricarda Schüler

10. März 2013

Zusammenfassung

Dieses Paper beschreibt das Aggregieren, Systematisieren und Vereinheitlichen von Fußballdaten verschiedener Quellen. Dazu werden Daten von Fussballdaten.de, Uefa.com, DBpedia, einem Twitter-Feed von Kicker.de und YouTube verwendet. Da eine Entität in verschiedenen Quellen eine unterschiedliche Repräsentation aufweisen kann, muss erkannt werden, welche Entitäten aus den verschiedenen Datenquellen einander entsprechen. Aufgrund der Tatsache, dass keine einheitlichen Formate oder Strukturen vorliegen, müssen verschiedene Zuordnungsalgorithmen angewandt werden, die das Zuordnen von Fußball-Entitäten einer Quelle zu einer anderen Quelle ermöglichen. In diesem Paper werden ebensolche Algorithmen beschrieben und evaluiert. Unsere Auswertungen zeigen, dass eine automatisierte Extraktion und Aggregation von Daten verschiedener Quellen in den meisten Fällen sehr gute Ergebnisse erzielt, aber dennoch fehleranfällig bei falschen oder inkonsistenten Quelldaten sein kann.

Inhaltsverzeichnis

1 Einleitung

Wenn man sich zum Thema Fußball informieren möchte, gibt es viele verschiedene Webseiten und Datenquellen. Man findet jedoch keine Quelle bei der alle verfügbaren Daten strukturiert zusammengetragen sind, sodass auf dieser Grundlage Statistiken erhoben werden können. Es gibt aber Situationen, in denen genau das interessant ist. Fragen wie „Welcher Spieler hat in all seinen Begegnungen die meisten Eigentore geschossen?“ sind nicht einfach zu beantworten, wenn man dafür viele verschiedene Datenquellen durchschauen muss. Zum Beantworten solcher Fragen werden sowohl aktuelle als auch historische Informationen benötigt. Deswegen ist es notwendig, verschiedene Datenquellen zusammenzuführen.
Ein Problem bei dem Zusammenführen von Daten verschiedener Datenquellen stellen allerdings unterschiedliche Datenformate dar. Die Daten können vollkommen unformatiert (freier Text), semiformatiert (z B. valides HTML) oder strukturiert (Datenformate wie JSON oder XML) sein. Die Daten müssen extrahiert und analysiert werden, um sie dann in strukturierter Form speichern zu können.
Ein wichtige Aufgaben ist dabei das Erkennen von Entitäten. Entitäten sind eindeutig identifizierende Teile der Domäne, wie zum Beispiel ein Spieler, eine Begegnung oder ein Team. Beim Zusammenfügen der unterschiedlichen Datenquellen müssen die Entitäten erkannt werden, um zum Beispiel das Eigentor der einen Datenquelle mit dem Spielernamen aus der anderen Datenquelle zu verknüpfen. Dabei stellen die unterschiedlichen Repräsentationen der Entitäten auf den jeweiligen Datenquellen ein Problem dar. So werden in manchen Fällen lediglich Künstlernamen gespeichert, in anderen Fällen gibt es Tippfehler oder landestypische Sonderzeichen werden anders behandelt.
In der nachfolgenden Arbeit wird dargelegt, wie diese Probleme gelöst werden können. Es wird vorgestellt, wie Fußballinformationen aus mehreren Datenquellen in einem System vereint und die einzelnen Entitäten erkannt werden können.

Kapitel **??** zeigt verwandte Arbeiten. In Kapitel **??** werden die verwendeten Technologien vorgestellt. Kapitel **??** zeigt den Ablauf des erarbeiteten Programms. In Kapitel **??** werden die verwendeten Datenquellen genauer beschrieben und erklärt wie aus ihnen Informationen gewonnen wurden. Im Anschluss werden die einzelnen Zuordnungsalgorithmen erläutert. Dazu gehört das Zusammenfügen der Daten von Fussballdaten.de und Uefa.com (Kapitel **??**) und das Erweitern unserer Daten mit Informationen von DBpedia (Kapitel **??**). Des Weiteren werden die Algorithmen zum Analysieren von Kicker-Tweets (Kapitel **??**) und der Aufbau des Schedulers (Kapitel **??**) erklärt. Anschließend folgt in Kapitel **??** eine Beurteilung und Diskussion der zuvor vorgestellten Algorithmen. Zuletzt fasst Kapitel **??** die Ergebnisse zusammen und gibt einen Ausblick auf eine mögliche Weiterführung des Projekts.

2 Wissenschaftliche Grundlagen und verwandte Arbeiten

Ein Großteil der Forschung zur Extraktion von Sportergebnissen und -daten beschäftigt sich entweder mit nur einer Datenquelle oder ausschließlich mit nicht-live-Daten. Hierbei

steht besonders die Named Entity Recognition (NER), also das Erkennen von Entitäten in einem freiem Text, im Vordergrund.
So versuchen [**?**] beispielsweise, die Anzahl der während eines Fußballspiels verfassten Tweets von Twitter zu analysieren, um somit auf Ereignisse wie Tore oder Auswechslungen zu schließen. Dabei verlassen sie sich beim Erkennen von Ereignissen ausschließlich auf die Anzahl der Tweets, die zu einem Zeitpunkt verfasst werden. Somit können keine in kurzen Zeitabständen ablaufenden Ereignisse erfasst werden, da diese als ein einziges Ereignis erkannt werden würden. Ähnlich gehen auch [**?**] bei der Erkennung von Ereignissen in Cricket-Spielen vor. Sie analysieren jedoch zusätzlich zur Anzahl der Tweets auch deren Inhalt, um die verfassten Tweets zu klassifizieren und somit Ereignisse mit höherer Genauigkeit zu erkennen.
Im Gegensatz dazu betrachten [**?**] die Datenbanken der FIFA zu den Finalrunden der Weltmeisterschaften 1990 und 1994, um aus diesen Daten einen Zusammenhang zwischen Passsequenzen und Toren zu erforschen. [**?**] untersuchen auf den Webseiten Uefa.com und Fifa.com Verfahren, um Daten aus unstrukturierten Texten mittels linguistischer Datenverarbeitung zu extrahieren. Einen ähnlichen Ansatz, der jedoch auf dem Erkennen von Mustern auf Webseiten zu Sportmeldungen basiert, verfolgen [**?**].
Während diese Verfahren gute Ergebnisse erzielen, können bei Analysen von Tweets keine historischen Daten berücksichtigt werden und bei der Verarbeitung von Texten zu Sportmeldungen keine Live-Daten. Verfahren, welche sich nur auf nutzergenerierten Inhalten verlassen ([**?**] und [**?**]), sind fehleranfällig, da der zugrundeliegende Datensatz unstrukturiert und vielfältig ist. Auch Ansätze, die ausschließlich auf der Named Entity Recognition oder der Erkennung von Textmustern beruhen ([**?**] und [**?**]), können je nach Textquelle in der Genauigkeit variieren.
In dieser Arbeit wird eine Möglichkeit vorgestellt, wie eine Kombination von sowohl Live- als auch historischen Daten zu Fußballspielen und -spielern gesammelt werden kann, um einen möglichst vollständigen und fehlerfreien Datensatz zu erhalten. Hierzu wird ein sich selbst aktualisierender Grunddatensatz erstellt, in dem historische Daten von Webseiten gesammelt werden, welche diese Informationen in Form von strukturierten Tabellen bereitstellen. Diese Quellen zeichnen sich wegen der manuellen Einpflegung der Daten sowohl durch ihre Vollständigkeit als auch durch ihre Korrektheit aus. Somit sind die Basisdaten verlässlich und die anschließende Named Entity Recognition der Live-Daten von Freitext-Tweets kann mit höherer Genauigkeit durchgeführt werden.

3 Erstellung des Fußball-Datensatzes

3.1 Verwendete Technologien

Als Datenbank für die extrahierten Daten der verschiedenen Webseiten wird der Triple-Store Virtuoso verwendet. Die Daten werden dabei als RDF-Tripel abgespeichert [**?**]. RDF (Resource Description Framework) ist eine gängige Sprache zum Beschreiben strukturierter Informationen. Vor allem Metadaten zu Informationen im Internet liegen oft im RDF-Format vor. Deshalb kann RDF als grundlegende Technologie des Semantic Webs, in dem Beschreibung von Informationen für Computer lesbar vorliegen, angesehen werden [**?**]. RDF wird auch als „gerichtetes, beschriftetes Graphdatenformat“ bezeichnet [**?**].

Es schreibt keine feste Struktur vor und kann leicht erweitert werden. Die Daten werden dabei als Tripel geschrieben, das heißt, es gibt ein Subjekt, ein Prädikat und ein Objekt [**?**]. In der Listing **??** ist ein Beispieltripel zu sehen, in dem definiert wird, dass Bastian Schweinsteiger ein Fußballspieler ist.

```
smm:Bastian_Schweinsteiger_1_08_1984 rdf:type smm:SoccerPlayer
```

Listing 1: Beispieltripel für „Bastian Schweinsteiger ist ein Fußballspieler“

Anfragen an den Triple-Store Virtuoso erfolgen mittels der speziell für RDF-Daten konzipierten Anfragesprache SPARQL. In Listing **??** ist eine Beispielanfrage zu sehen, in welcher alle Fußballspieler aus der Datenbank abgefragt werden. Das Ergebnis ist eine Menge von URIs (Uniform Resource Identifier), die jeweils in der Variablen `s` gebunden sind. Eine URI (Uniform Resource Identifier) ist ein eindeutiger Bezeichner einer Ressource. In der Anfrage werden sogenannte Prefixe festgelegt, welche als Abkürzung dienen. So steht innerhalb der Anfrage die Abkürzung `smm` für den Ausdruck `http://purl.org/hpi/soccer-voc/`.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX smm: <http://purl.org/hpi/soccer-voc/>

SELECT *
WHERE
{
  ?s rdf:type smm:SoccerPlayer.
}
```

Listing 2: Beispielanfrage um alle Fußballspieler in der Datenbank abzufragen

3.2 Genereller Ablauf

Der allgemeine Ablauf des Programmes lässt sich in vier Phasen unterteilen:

1. Die Daten der einzelnen Seiten werden einmalig heruntergeladen und lokal gespeichert, damit keine überflüssigen Anfragen an die einzelnen Webseiten gestellt werden müssen. Die heruntergeladenen Daten sind reine Rohdaten, also komplett unverarbeitet. Im Beispiel von Fussballdaten.de sind dies die reinen HTML-Seiten.

2. Zunächst werden die Daten von der Quelle Fussballdaten.de geparst. Dazu werden alle zuvor heruntergeladenen Dateien dieser Quelle wie folgt verarbeitet:

   - Für jede heruntergeladene Datei werden die entsprechenden Daten geparst.
   - Die extrahierten Daten werden zunächst in entsprechende Modellklassen (z. B. *SoccerMatch*, *Player* oder *Goal*) gespeichert.
   - Nach dem Parsen jeder Datei werden RDF-Tripel erzeugt. Jede Modellklasse kennt dabei die Struktur der Tripel, die sie erzeugen muss. Die RDF-Tripel werden direkt in eine Datei geschrieben, um sie nicht zwischenspeichern zu müssen.

Nachdem alle Daten der Quelle Fussballdaten.de geparst wurden, werden alle Tripel in einem Bulk-Load in den Triple-Store geschrieben.

3. Anschließend werden die Daten von Uefa.com geparst und als Tripel in der Datenbank gespeichert. Hierbei muss es zu einer Zuordnung zu den bereits in der Datenbank vorhanden Entitäten kommen, da sonst doppelte Spieler gespeichert werden würden.

4. Danach werden eine Reihe von nachgelagerten Datenanreicherungen durchgeführt. Dies sind Operationen, welche dazu dienen, vorhandene Entitäten mit weiteren Eigenschaften aufzufüllen. Dazu gehört u.a. das Zuordnen der Daten zu DBpedia-Entitäten und das Analysieren von Kicker-Tweets.

Zum Präsentieren der entstandenen Datenmenge wurde eine Webseite erstellt. Auf dieser kann man alle bekannten Informationen zu den einzelnen Entitäten einsehen. Weiterhin wurden Statistiken erzeugt, die die Vielfalt des Datensatzes demonstrieren. Drei beispielhafte Anfragen dazu finden sich in Anhang **??**.

3.3 Parsing der einzelnen Datenquellen

Um eine möglichst große Menge an Daten zusammenzutragen, werden die folgenden fünf Datenquellen verwendet:

- Fussballdaten.de[1]
- Uefa.com[2]
- DBpedia[3]
- Kicker-Feed auf Twitter[4]
- Sky Sport HD YouTube-Kanal[5]

Die Daten, die von diesen Quellen geparst werden, werden in Form von RDF-Tripeln nach dem in Abbildung **??** dargestellten Schema in den Triple-Store Virtuoso eingefügt. Fussballdaten.de dient dabei als Hauptquelle, das heißt, die von hier bezogenen Daten werden als korrekt angenommen. Innerhalb der Fussballdaten.de-Daten findet keine Zuordnung statt, da eine Entität auf der Seite immer in der gleichen Schreibweise auftaucht. Gleichzeitig ist Fußballdaten.de die umfangreichste Datenquelle. Die Daten in dieser Quelle sind strukturiert und nach einem festem Schema aufgebaut, außerdem werden sie manuell gepflegt.

[1] `www.fussballdaten.de`
[2] `http://www.uefa.com`
[3] `http://de.DBpedia.org`
[4] `twitter.com/kicker_bl_li`
[5] `http://www.youtube.com/user/SkySportHD`

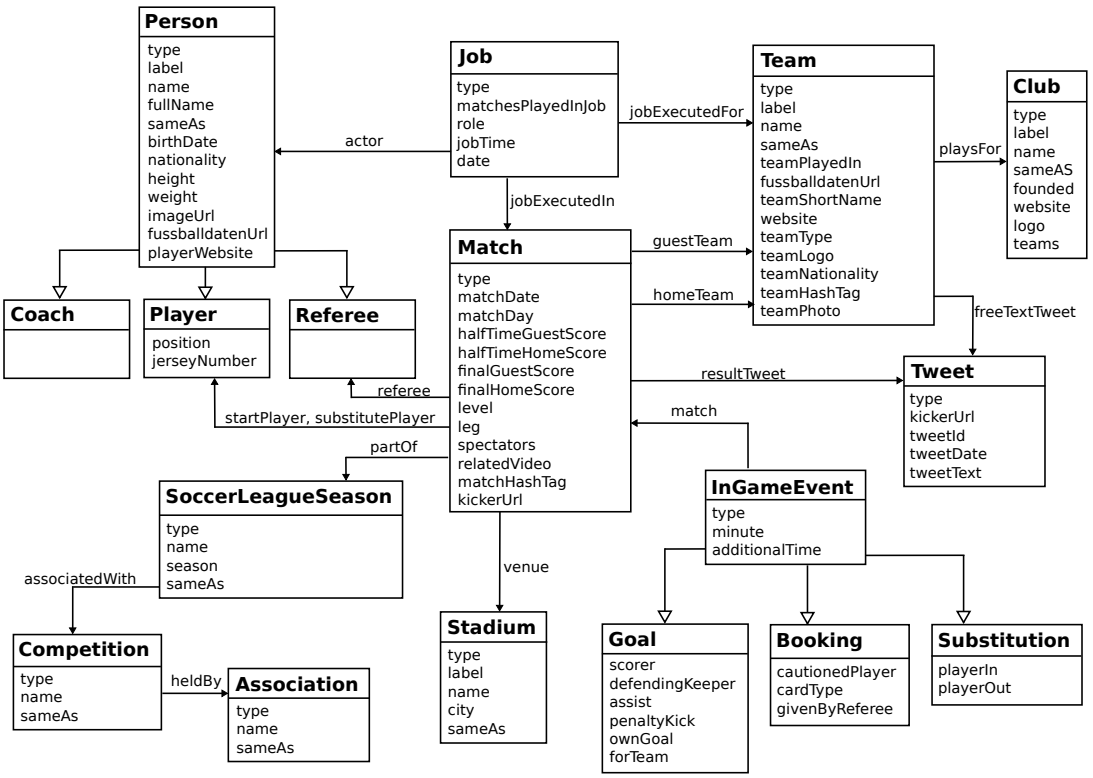

Abbildung 1: Datenschema im Virtuoso Tripel-Store für alle Datenquellen

3.3.1 Fussballdaten.de

Von dieser Quelle werden Spieler und Mannschaften sowie alle Begegnungen der 1. und 2. Bundesliga, der WM und der EM geparst. Fussballdaten.de zeichnet sich dadurch aus, dass es für Begegnungen, Spieler und Mannschaften jeweils eigene Seiten gibt, die umfangreiche Daten enthalten.
Um diese Daten zu erhalten, muss zunächst einmal festgestellt werden, welche URL die spezifischen Seiten haben. Für die Mannschaften und Spieler ist dies zentral möglich, da es jeweils eine Seite gibt, auf denen sowohl alle Spieler als auch alle Mannschaften mit einem Link zu ihrer spezifischen Seite aufgelistet sind. Da eine solche Auflistungsseite jedoch nicht für Begegnungen existiert, muss das Schema, nach dem sich die URLs der Begegnungen zusammensetzen, mittels eines regulären Ausdrucks im Programm für alle Begegnungen aller Spieltage generiert werden. Sobald die genauen URLs der jeweiligen Seiten bekannt sind, können diese geparst werden. Dabei stehen die für das Datenschema relevante Daten immer in gleich aufgebauten Tabellen.

3.3.2 Uefa.com

Von dieser Webseite werden, ähnlich zu Fussballdaten.de, Informationen zu Spielern, Mannschaften und Begegnungen geparst. Diese Webseite dient jedoch nur als Informa-

tionsquelle für die Champions League. Für das Parsen der Daten wird ein lokal gespeicherter Dump – also ein Abbild der Daten von der Webseite – genutzt. Dieser besteht aus HTML-Seiten über alle Champions-League-Begegnungen, sowie einer JSON-Datei pro Begegnung, welche weiterführende Informationen zu allen beteiligten Personen beinhaltet.
Das eigentliche Parsing geschieht ausgehend von den HTML-Dateien der Begegnungen, welche zusätzlich zu den Datenschema-konformen Informationen wie Austragungsort oder Endergebnis noch eine Uefa-interne Begegnungs-ID enthalten. Diese wird benutzt, um die zu einer Begegnung gehörige JSON-Datei zu erhalten. In einer solchen JSON-Datei finden sich Informationen wie Geburtsdatum oder Gewicht zu denen an einer Begegnung beteiligten Schiedsrichtern, Trainern und Spielern.

3.3.3 DBpedia

DBpedia ist ein Projekt, das Daten von Wikipedia strukturiert zur Verfügung stellt. Dabei sind diese Daten in einem Virtuoso Tripel-Store abgelegt, welchen man über einen SPARQL-Endpunkt ansprechen kann. Durch die Verbindung von DBpedia zur Wikipedia handelt es sich bei der DBpedia um eine Datenquelle von großem Umfang. In ihr werden viele Informationen der Wikipedia-Artikel, wie die Infoboxen oder Tabellen, als RDF-Tripel gespeichert. Da diese Infoboxen jedoch keinerlei Standards unterliegen, variiert ihre Form sehr stark. Dies führt dazu, dass die DBpedia-Daten nicht immer zuverlässig sind. Im Konkreten bedeutet das, dass in manchen Fällen als Geburtsdatum nur die Zahl „1“ eingetragen ist, oder der Monat mit dem Tag vertauscht ist. Ein anderes Beispiel ist der Spitzname eines Spielers, der als Objekt nach den Prädikaten `dbprop:fullname`, `dbprop:name`, `dbprop:nickname` oder `rdfs:label` auftauchen kann.
DBpedia wird verwendet, um die bereits vorhandenen Entitäten mit weiteren Informationen aus der deutschen und englischen DBpedia anzureichern. Dazu wird sowohl der deutsche als auch der englischen SPARQL-Endpunkt der DBpedia angefragt.

3.3.4 Twitter

Bei dem Kicker Twitter-Feed handelt es sich um die Tweets der deutschen Sportzeitschrift „Kicker-Sportmagazin“. Das Besondere an dieser Datenquelle ist, dass sie mit Live-Daten von Bundesligaspielen aktualisiert wird, während die Begegnungen laufen. Diese Tweets sind in immer gleichbleibender Form und daher gut automatisiert auslesbar. Des Weiteren bietet sie aktuelle Hintergrundinformationen zu Bundesligavereinen in Form von Freitext-Tweets, die ebenfalls geparst und analysiert werden. Die Tweets werden genutzt, um unsere Daten auch während der Spiele aktuell zu halten.

3.3.5 YouTube.com

Als Datenquelle für Videos wurde der Kanal von *Sky Sport HD* auf YouTube.com genutzt. Hierbei wird im Programm die YouTube-API[6] angesprochen, um relevante Videos für jede Begegnung der 1. Bundesliga und Champions League ab der Saison 2011/2012

[6] `https://developers.google.com/youtube/2.0/developers_guide_protocol`

als Tripel zu der jeweiligen Begegnung zu erzeugen. Dafür wird der *Sky Sport HD*-Kanal nach Videos durchsucht, deren Titel die beiden Mannschaftsnamen sowie den Spieltag enthalten. Dabei werden nur Videos beachtet, welche in der Woche nach dem Spiel veröffentlicht wurden. Die zu einer Begegnung relevanten Videos werden dann auf der entsprechenden Seite der Webseite angezeigt.

3.4 Zuordnen der Uefa.com Datenquelle zur Fussballdaten.de Basisquelle

Um Informationen zu Begegnungen der Uefa Champions League zu erhalten, wurden die im Dump bereitgestellten Daten geparst. Aus den gewonnen Daten werden 22417 Spieler sowie 25 deutsche Mannschaften erlangt, welche sich potenziell schon im Virtuoso Triple-Store befinden. Durch die verwendete Basisdatenquelle Fussballdaten.de sind bereits sowohl 46701 internationale Spieler als auch 838 deutsche Vereinsmannschaften im Triple-Store abgelegt. Somit muss verhindert werden, dass Entitäten, welche bereits vorhanden sind, von Uefa.com als Duplikat hinzugefügt werden. Dazu ist es notwendig, alle von Uefa.com geparsten Spieler und Mannschaften mit den bereits vorhandenen Daten zu vergleichen und wenn nötig, Duplikate zusammenzufügen.

3.4.1 Vorgehensweise

Das Parsen von Uefa.com-Daten wird stets nach dem Verarbeiten der Basisdatenquelle Fussballdaten.de ausgeführt, wodurch sichergestellt ist, dass Stammdaten bereits vorhanden sind. Im Tripel-Store haben alle Spieler eine URI der Form

`http://purl.org/hpi/soccer-voc/Philipp_Lahm_1983-11-11`,

welche aus dem Namen und dem Geburtsdatum des jeweiligen Spielers generiert wird. Analog haben Mannschaften jeweils eine URI, welche jedoch nur aus dem Mannschaftsnamen gebildet wird. Werden mehrere Entitäten mit den gleichen URIs in den Tripel-Store geladen, werden diese als die gleiche Entität behandelt. Aus diesem Grund müssen beim Zuordnen nur die Schlüsselwerte betrachtet werden, aus welchen die URI aufgebaut ist. Folglich müssen, falls für einen Spieler von Uefa.com erkannt wird, dass sich dieser bereits in der Virtuoso Datenbank befindet, nur der Name und das Geburtsdatum des Uefa.com-Spielers angepasst werden, wodurch das Duplikat aufgelöst wird.
Hierbei wird davon ausgegangen, dass die Daten von Fussballdaten.de korrekt sind, da bei einer Stichprobe von 150 Uefa.com-Spielern bei 21 das angegebene Geburtsdatum vom tatsächlichen Datum abwich.
Nach dem Auflösen des Duplikats wird der Spieler mit veränderten, also von Fussballdaten.de übernommenen, Werten in die Datenbank geladen. Wird der Uefa-Spieler nicht in der Datenbank gefunden, so wird er unverändert in der Datenbank abgelegt, wodurch der Spieler neu hinzugefügt wird. Für Mannschaften wird analog vorgegangen.

3.4.2 Zuordnung der Spieler

Jeder Spieler, der von Uefa.com geparst wird, wird mit der Virtuoso Datenbank verglichen, um Duplikate zu verhindern (siehe Abbildung **??**). Dazu wird für den jeweiligen

Uefa.com-Spieler eine SPARQL-Anfrage gestellt, welche eine Ergebnismenge zurückgibt, die alle Spieler mit gleichem Geburtsdatum und alle Spieler mit gleichem Namen beziehungsweise gleichem normalisierten Namen vereinigt. Somit ist es möglich, die Spieler zu ermitteln und dem Grunddatensatz zuzuordnen, welche entweder ein bezüglich der Basisdaten falsches Geburtsdatum oder einen abweichenden Namen haben.
Befindet sich in der Ergebnismenge nur ein Spieler, stimmen entweder sein Name oder sein Geburtsdatum genau überein. Bei Übereinstimmung des Geburtsdatums wird geprüft, ob die Levenshtein-Differenz der beiden Namen kleiner als drei ist. Ist dies der Fall, wird der Spieler zugeordnet. Stimmt der Name genau überein, wird, selbst bei unterschiedlichen Geburtsdaten, angenommen, dass es sich um den gesuchten Spieler handelt, da die Geburtsdaten einiger Uefa-Spieler fehlerhaft sind (siehe Abschnitt **??**).

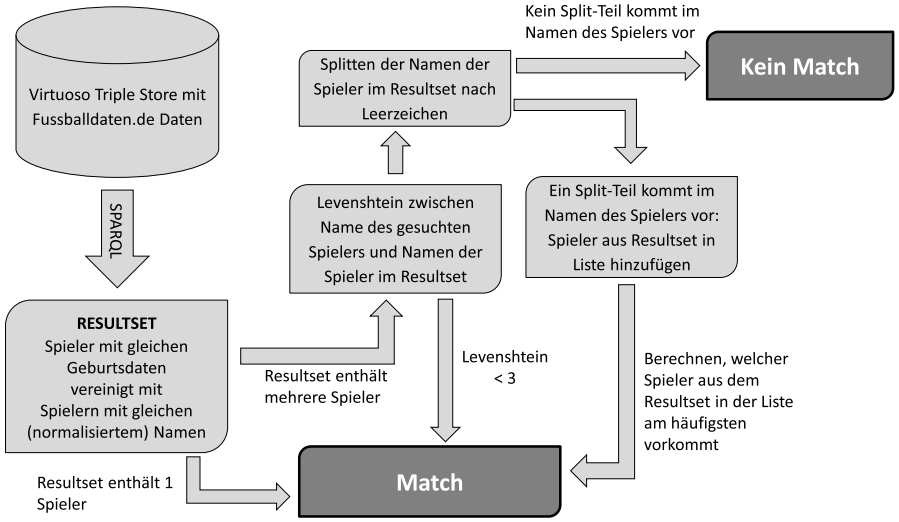

Abbildung 2: Zuordnen der Spieler von Uefa.com zu Fussballdaten.de

Erzielte die SPARQL-Anfrage mehr als ein Ergebnis, so wird für jeden Spieler in der Ergebnismenge die Levenshtein-Distanz zum Namen des Uefa.com-Spielers berechnet. Die Levenshtein-Distanz kann als Maß für die Ähnlichkeit zweier Zeichenketten verwendet werden. Dabei wird die Anzahl der Operationen berechnet, die benötigt werden um eine Zeichenkette in eine Zweite zu überführen. Eine Operation besteht dabei aus Ersetzungen, Löschungen und Einfügungen einzelner Zeichen in die initiale Zeichenkette, solange bis die Zielzeichenkette konstruiert wurde [**?**]. Ist die Distanz dieser Berechnung kleiner als drei, wird davon ausgegangen, dass der Uefa.com-Spielername falsch geschrieben wurde. Folglich wird wiederum der Name und falls nötig, das Geburtsdatum angepasst. Beispielsweise kann „Yegor Titov“ so zu dem bei Fussballdaten.de als „Jegor Titow“ eingetragenen Spieler zugeordnet werden.
Konnte mittels des Levenshtein-Algorithmus kein Treffer erzielt werden, wird untersucht, ob das Zuordnen daran scheitert, dass zu einem Spieler in einer der beiden Datenquellen mehrere Namen gespeichert sind. Beispielsweise ist „António Henrique Jesus Oliveira“ bei Fussballdaten.de nur als „Antonio Oliveira“ eingetragen und könnte somit durch ledigliche Überprüfung des vollen Namens nicht zugeordnet werden. Deshalb wird der

gesuchte Uefa.com-Spieler vorerst nach Leerzeichen und Bindestrichen zerlegt. Für jeden Namensteil wird anschließend geprüft, ob er in einem Namen eines Spielers in der Ergebnismenge enthalten ist. Für jedes Vorkommen des Namensteils wird der gefundene Spieler aus der Ergebnismenge einer Liste hinzugefügt. Um den Spieler mit größter Übereinstimmung zu ermitteln, wird der Spieler, welcher in der Liste am häufigsten vorkommt zurückgegeben. Dieser Spieler ist gleichzeitig der, für den die meisten Namensteile mit dem des Uefa.com-Spielers übereinstimmen und welcher verwendet wird um Namen und Geburtsdatum des Uefa.com-Spielers anzupassen.
Schlagen alle Überprüfungen fehl, konnte der Spieler von Uefa.com nicht zur Basisdatenquelle zugeordnet werden.

3.4.3 Zuordnung der Mannschaften

Um das Problem der Duplikaterkennung bei Mannschaften zu bewältigen, wird der Mannschaftsname nach Leerzeichen zerlegt und eine SPARQL-Anfrage gebaut, die alle Mannschaften zurückgibt, welche einen Teil des Mannschaftsnamens der gesuchten Mannschaft enthält. Anschließend wird die Levenshtein-Distanz auf dem Gesamtnamen der gesuchten Mannschaft und den Namen der jeweiligen Mannschaft in der Ergebnismenge berechnet. Ist diese kleiner als drei, so wird der Mannschaftsname der gesuchten Mannschaft an den gefundenen Namen angepasst.
Konnte so keine Zuordnung ermittelt werden, wird der Name der gesuchten Mannschaft ein weiteres Mal nach Leerzeichen zerlegt. Nachfolgend wird geprüft, ob die Mannschaftsnamen in der Ergebnismenge einen Teil des zerlegten Namens enthalten. Dabei wird darauf geachtet, dass der Namensteil größer als drei ist, damit Zeichenketten wie „FC“ nicht zugeordnet werden. Die jeweils gefundene Mannschaft ist die, in der die meisten Namensteile enthalten sind.
Waren keine Namensteile enthalten, so kann die Mannschaft nicht zugeordnet werden.

3.5 Zuordnung von DBpedia-Entitäten

Die Zuordnung von DBpedia-Entitäten auf unsere Entitäten wird vollzogen, um unseren Datenbestand mit weiteren Daten anzureichern. Dazu muss zunächst festgestellt werden, welche DBpedia-Entität unserer Entität entspricht. Im Folgenden wird auf den Algorithmus dieser Zuordnung eingegangen. Dabei ist zu beachten, dass sich dieser Algorithmus unterscheidet, je nach dem welche Art von Entität betrachtet wird. Da die Algorithmen für die Zuordnung von Spielern und Mannschaften die Grundlegendsten sind, wird auf diese besonders eingegangen.

3.5.1 Zuordnung der Spieler

Der Algorithmus zum Zuordnen eines Spielers der DBpedia wird anhand des Beispiels des Spielers „Bastian Schweinsteiger“ vorgestellt.
Grob kann man den Algorithmus in zwei Teile gliedern. Im ersten Teil wird die DBpedia nach möglichen passenden Spielern angefragt, im Zweiten wird geprüft, welcher dieser möglichen Spieler wirklich mit dem Spieler übereinstimmt, nach dem gesucht wurde.

Die Anfrage an die DBpedia basiert auf dem Namen des Spielers. Der prinzipielle Aufbau der Anfrage wird in Listing **??** gezeigt.

Listing 3: DBpedia Anfrage für Spieler

```
SELECT DISTINCT * WHERE {
  { ?url <http://dbpedia.org/property/name> ?name .
    ?name <bif:contains> '<Spielername>' . }
  UNION
  { ?url <http://xmlns.com/foaf/0.1/name> ?name .
    ?name <bif:contains> '<Spielername>' . }
  UNION
  { ?url <http://www.w3.org/2000/01/rdf-schema#label> ?name .
    ?name <bif:contains> '<Spielername>' . }
}
```

'<Spielername>' ist dabei nur ein Platzhalter für den richtigen Namen des Spielers. Wie in Listing **??** zu sehen ist, muss der Name des gesuchten Spielers in dem Objekt eines der drei Tripel `rdf:label`, `dbpprop:name` oder `foaf:name` enthalten sein. Das Überprüfen, ob der Name in einen dieser Tripel vorkommt, geschieht mittels einem `<bif:contains>`. Diese boolesche Operation basiert auf einem Freitextindex [**?**] und ist dadurch schneller als eine Filterung mittels regulärer Ausdrücke. Aus dem Ergebnis der Anfrage wird die DBpedia-URI und alle Objekte der angefragten Tripel, welche verschiedene Varianten des DBpedia-Namen darstellen, gespeichert.
Eine Besonderheit des Algorithmus ist, dass der Name des Spielers vor dem Stellen der Anfrage anhand des Leerzeichen getrennt wird. Die endgültige Anfrage besteht dann aus mehreren Teilen, die miteinander vereinigt werden. Ein Teil ist dabei die in Listing **??** dargestellte WHERE-Klausel. In jedem dieser Teile wird dabei für den Platzhalter '<Spielername>' der entsprechende Teilname eingesetzt.
Bei unserem Beispiel wird der Name des Spielers „Bastian Schweinsteiger" in zwei Teile aufgeteilt: „Bastian" und „Schweinsteiger". Die Anfrage sieht dann folgendermaßen aus:

Listing 4: DBpedia Anfrage für Bastian Schweinsteiger

```
SELECT DISTINCT ?url ?name
WHERE {
  { ?url <http://dbpedia.org/property/name> ?name .
    ?name <bif:contains> "Bastian" .
  }
  UNION
  { ?url <http://xmlns.com/foaf/0.1/name> ?name .
    ?name <bif:contains> "Bastian" .
  }
  UNION
  { ?url <http://www.w3.org/2000/01/rdf-schema#label> ?name .
    ?name <bif:contains> "Bastian" .
  }
  UNION
  { ?url <http://dbpedia.org/property/name> ?name .
    ?name <bif:contains> "Schweinsteiger" .
  }
  UNION
  { ?url <http://xmlns.com/foaf/0.1/name> ?name .
```

```
    ?name <bif:contains> "Schweinsteiger" .
  }
  UNION
  { ?url <http://www.w3.org/2000/01/rdf-schema#label> ?name .
    ?name <bif:contains> "Schweinsteiger" .
  }
}
```

Die Namenstrennung wird vorgenommen, da DBpedia keine unscharfe Suche ermöglicht. So wird zum Beispiel der Fußballspieler „Thomas Müller“ gefunden, obwohl dieser Spieler unter Umständen in der DBpedia nur unter „Thomas Muller“ abgespeichert ist.
Durch die Teilung des Namens kann es passieren, dass die Ergebnismenge der SPARQL-Anfrage, welche höchstens 40000 Zeilen beinhalten darf, zu groß wird [**?**]. In einem solchen Fall wird der Fehler `HttpException: 500 SPARQL Request Failed` geworfen. Um diesen Fehler zu vermeiden wird zunächst versucht die Anfrage mit dem getrennten Namen zu stellen. Wird hierbei ein Fehler geworfen, wird die Anfrage nochmals gestellt, diesmal jedoch mit dem vollen Namen des Spielers, wodurch die Anzahl der abgefragten Spielern verringert wird.
Im Anschluss werden alle gefunden DBpedia-Seiten gefiltert. Dazu werden alle Namen, die zu der DBpedia-URI gespeichert wurden, mittels der Levenshtein-Distanz mit dem Namen, den wir für diesen Spieler gespeichert haben, verglichen. Ist eine dieser Levenshtein-Distanzen kleiner als zwei, wird die entsprechende URI im Algorithmus weiter betrachtet. Diese strenge Aussortierung wird vorgenommen, um Seiten wie zum Beispiel `http://DBpedia.org/page/Tobias_Schweinsteiger` beim Suchen nach „Bastian Schweinsteiger“ aus der Ergebnismenge zu filtern. Da der DBpedia-Name fast identisch mit dem in unserer Datenbank befindlichen Namen ist, werden nach dieser Aussortierung nur noch Seiten betrachtet, welche mit hoher Wahrscheinlichkeit ein Treffer sind.
Der nächste Schritt ist nun, die Korrektheit der noch verbliebenen DBpedia-Seiten zu überprüfen. Dazu werden alle Tripel, die zu der DBpedia-URI gehören, heruntergeladen. Ist in den Tripeln ein *Redirect* (`dbpo:wikiPageRedirects`) oder ein *Disambiguate* (`dbpo: wikiPageDisambiguates`) enthalten, so werden diese verfolgt und die weitergeleiteten Seiten als mögliche passende Seiten betrachtet.
Damit die DBpedia-Seite als korrekte Zuordnung zu einem Spieler zählt, müssen folgende Kriterien erfüllt sein:

- Es darf sich bei der DBpedia-Seite nicht um einen Verein handeln. Dies ist der Fall, wenn der `rdf:type` der DBpedia-Entität keine `dbpo:Organisation` oder kein `dbpo:SoccerClub` ist.
- Das Geburtsdatum der DBpedia-Entität muss mit dem Geburtsdatum des gesuchten Spielers übereinstimmen. Liegt in unserer Datenbank oder bei der DBpedia kein Geburtsdatum vor, wird dieses Kriterium nicht berücksichtigt. Für das Geburtsdatum der DBpedia-Entität werden die Tripel `dbprop:birthDate` und `DBpedia-owl:birthDate` betrachtet.
- Die Entität der DBpedia-Seite muss einem Fußballspieler entsprechen. Dies wird überprüft, indem im `dbpo:abstract` oder im `rdfs:comment` eines der Wörter

„footballer“, „player“, „goalkeeper“, „defender“, „striker“, „midfielder“, „torwart“, „torhüter“, „stürmer“, „mittelfeld“, „abwehr“ oder „spieler“ enthalten ist. Zusätzlich sollte im `dbpo:abstract` oder im `rdfs:comment` das Wort „fußball“ oder „football“ vorkommen. Auf die Klein- und Großschreibung wird bei der Prüfung nicht geachtet.

Sind alle Kriterien erfüllt, wird zu der gesuchten Spieler-Entität ein `owl:sameAs` Tripel mit der entsprechenden DBpedia-URI angelegt und die DBpedia-Daten werden in die Datenbank geladen.
Der Algorithmus für Schiedsrichter und Trainer ist analog zu dem des Spielers.

3.5.2 Zuordnung der Mannschaften

Der Algorithmus zum Zuordnen von Mannschaften lässt sich in drei Schritte unterteilen:

1. Auf Basis des Fussballdaten.de-Namens einer Mannschaft werden mögliche Namen der Mannschafts-Entitäten auf DBpedia generiert. Dazu gehört, dass Leerzeichen konsistent zur DBpedia enkodiert werden (Fussballdaten.de verwendet teilweise keine Standard-Leerzeichen) und mögliche Sprachkennungen (`@en` oder `@de`) angehangen werden.

2. Anschließend wird eine Anfrage an die DBpedia gestellt, um alle Entitäten, deren Label (`rdf:label`) den zuvor generierten Mannschaftsnamen enthält, abzufragen.

3. Die Ergebnisse der Anfrage werden anschließend gefiltert: Manche zurückgegebenen Entitäten sind lediglich Kategorien (Entitäten, die Obergruppe für eine Menge von anderen Entitäten sind), oder betreffen Frauen-Mannschaften oder Mannschaften anderer Sportarten. Deswegen werden Entitäten, die Begriffe wie ”Kategorie”, ”Women”, ”Basketball”, ”Handballënthalten, aussortiert.

Der Algorithmus zum Zuordnen von Vereinen ist analog zu dem von Mannschaften.

3.5.3 Zuordnung weiterer Entitäten

Neben der schon beschriebenen Zuordnung von Personen und Mannschaften auf DBpedia-Entitäten werden in unserem Programm auch Stadien, Wettbewerbe, Saisons und Verbände zugeordnet.

- Die Zuordnung der Stadien erfolgt ähnlich wie das Zuordnen der Spieler. Es wird eine Anfrage an die DBpedia gestellt, in welcher der Name des Stadions im Objekt eines der folgenden Tripel vorkommen muss: `rdf:label`, `dbprop:name`, `dbprop:formerNames`, `dbprop:stadiumName` oder `foaf:name`. Danach werden die DBpedia-URIs mittels der Levenshtein-Distanz gefilter. Es werden nur noch die DBpedia-URIs betrachtet, deren Levenshtein-Distanz kleiner als vier ist. Anschließend wird überprüft, ob es sich bei der DBpedia-Entität um ein Stadion handelt. Dies ist der Fall wenn im `dbpo:abstract` oder im `rdfs:comment` das Wort „stadium“ oder „stadion“ vorkommt. Die DBpedia-Entität wird dann als Treffer gewertet.

- Es gibt insgesamt fünf verschiedene Wettbewerbe und sieben Verbände. Die URIs zu den DBpedia-Entitäten für diese Wettbewerbe und Verbände sind sehr unterschiedlich aufgebaut. Aus diesen Gründen werden Wettbewerbe und Verbände per Hand den richtigen DBpedia-Entitäten zugeordnet.

- Die URIs der Saisons in der DBpedia weisen ein eindeutiges und einheitliches Schema auf. Daher werden ausgehend von unseren Saisons automatisiert die DBpedia-URIs für die einzelnen Saisons erzeugt, sodass dadurch eine Zuordnung unserer Entitäten auf die DBpedia erfolgt.

3.6 Twitter

Die Daten von Fussballdaten.de werden ungefähr eine Stunde nach Ende einer Begegnung aktualisiert. Um auch während einer Begegnung aktuelle Spieldaten zu erhalten, wird ein Twitter-Feed verwendet, der Echtzeit-Spielergebnisse bereitstellt[7]. Dieser wird von *Kicker*, einer deutschen Fußballfachzeitschrift, bereitgestellt, was von Twitter offiziell verifiziert wird.
Im diesem Twitter-Feed werden nur Tweets über die Bundesliga veröffentlicht; es gibt jedoch alternative Benutzerkonten wie `kicker_2_bl_li` (2. Bundesliga), `kicker_cl_li` (Champions League) oder `kicker_em_li` (Europameisterschaft), die auch zu anderen Wettbewerben Live-Daten bereitstellen. Im Folgenden werden jedoch nur Bundesliga-Tweets betrachtet.
Die Tweets dieses Nutzers lassen sich in zwei Kategorien einteilen:

- *Ergebnis-Tweets*: Hierbei handelt es sich um Tweets, die den Ergebniszwischenstand einer Begegnung mitteilen. Dabei wird für jedes Tor ein Tweet veröffentlicht, ebenso zu Spielbeginn, zum Ende der 1. Halbzeit, zu Beginn der 2. Halbzeit und nach Spielende.

  Die Tweets zu einem Tor haben ein festes Format, welches die beiden Mannschaften, den aktuellen Spielstand, den Torschützen und die Minute beinhaltet. Abbildung **??** zeigt einen solchen Tweet. Aufgrund der großen Anzahl verfasster Tweets

  Bayern München - FC Schalke 04 4:0 Tor:
  Gomez (63., Robben) #FCBS04
  http://bit.ly/Yoh4TD

  Abbildung 3: Beispiel für einen Ergebnis-Tweet des Kicker-Twitter-Feeds

  zu einzelnen Spielen liegt die Vermutung nahe, dass diese Tweets automatisch generiert werden.

- *Freitext-Tweets*: Zusätzlich zu den Tormeldungen gibt es Tweets, die keinem festen Schema folgen, fortan als „Freitext-Tweets“ bezeichnet. Diese enthalten Hintergrundberichte zu aktuellen Bundesliga-Mannschaften, wie zum Beispiel Verletzungen, Vereinswechsel, Interviews oder Sperren.

[7] `https://twitter.com/kicker_bl_li`

Ziel ist es, bei diesen Tweets die Mannschaften zu erkennen, von denen der Tweet handelt, und diese Beziehung zwischen Tweet und Mannschaft zu speichern.

3.6.1 Bearbeiten der Twitter-Feeds

Die Bearbeitung des Twitter-Feeds erfolgt in Phase **??** (siehe Kapitel **??** über den allgemeinen Ablauf). Dabei werden alle Tweets, die auf Festplatte geschrieben wurden, eingelesen, und zusätzlich die Twitter-API nach den neuesten Tweets angefragt. Insgesamt gibt es 3804 Tweets seit Beginn des Twitter-Feeds am 10. März 2012. Davon waren 1677 Freitext-Tweets und 2127 Ergebnis-Tweets.
Für jeden dieser Tweets wird eine Funktion aufgerufen, die die passenden Tripel für diesen Tweet erzeugt. Dafür muss die Funktion zunächst entscheiden, ob es sich um einen Freitext-Tweet oder einen Ergebnis-Tweet handelt. Dies geschieht mittels eines regulären Ausdrucks, der überprüft, ob die feste Form eines Ergebnis-Tweets gegeben ist. Dabei wird zwischen einem Tor-Tweet, einem Halbzeit-Tweet und einem Schlusspfiff-Tweet unterschieden. Handelt es sich nicht um einen Ergebnis-Tweet, wird von einem Freitext-Tweet ausgegangen. Aufgrund der festen Form funktioniert dies zuverlässig. Im Statistik-Anhang wird eine Stichprobe von Tweets mit den vom Algorithmus erkannten Tweet-Typ aufgezeigt. Die Tweets wurden dabei zufällig ausgewählt. Die Untersuchung der Stichprobe der Größe 300 ergab, dass die Zuordnung bei allen Tweets korrekt funktionierte. Damit liegt die Genauigkeit der korrekten Zuordnung der Tweets zu 95 % über 98,74 %.
Handelt es sich um einen Ergebnis-Tweet, ist eine weitere Unterscheidung notwendig: die Kategorisierung nach Halbzeit, Schlusspfiff oder Tor. In den ersten beiden Fällen werden Tripel erzeugt, die die Existenz des Spiels bezeugen, das heißt es wird eine Begegnungsentität mit den beiden teilnehmenden Mannschaften erzeugt, das Datum vermerkt und der entsprechende Kicker-Link gespeichert. Handelt es sich um einen Ergebnis-Tweet mit Tormeldung werden zusätzlich die Tripel für das Tor und den Torschützen erzeugt. Wie die Erzeugung der Tripel funktioniert, wird im Folgenden beschrieben. Dabei wird je nachdem, um welche Tweet-Typ es sich handelt, ein anderer Extraktions-Algorithmus angewandt.

3.6.2 Parsen von Ergebnis-Tweets

Aus einem Ergebnis-Tweet können die Daten zu den beiden teilnehmenden Mannschaften, das Ergebnis, den Torschützen, den Vorbereiter (nicht immer vorhanden), die Minute, das Datum, den Hashtag der Begegnung und die URL zu einer detaillierteren Kicker-Seite mithilfe eines regulären Ausdrucks extrahiert werden.
Es verbleibt, die beiden Zeichenketten für die Mannschaften und die beiden Spieler (Torschütze, Vorbereiter) den entsprechenden Entitäten in der Datenbank zuzuordnen.
Dies geschieht unter der Annahme, dass die entsprechenden Spieler und Mannschaften in der Datenbank bereits existieren. Da unsere Architektur vorsieht, dass vor dem Bearbeiten des Twitter-Feeds bereits das Parsen von Fussballdaten.de abgeschlossen ist, ist dies stets gegeben. Ein Problem besteht lediglich für neue Entitäten. Für Mannschaften ist dies wiederum kein Problem, da alle Mannschaften aus der Bundesliga bekannt sind.

Wenn allerdings ein Spieler in seinem ersten Bundesligaspiel ein Tor schießt, wird der Algorithmus keinen Spieler finden. Da später am gleichen Tag (siehe Kapitel **??** zum Scheduler) die aktuellen Daten von Fussballdaten.de geparst werden, wird diese Informationslücke aber schnell geschlossen. Abbildung **??** zeigt die Anzeige des Spielstandes basierend auf den Tweet-Daten während eines Spiels.

Abbildung 4: Live-Anzeige des Spielstandes auf der Webseite während einer Begegnung

Beim Zuordnen der Mannschaften geht es darum, Zeichenketten wie beispielsweise „Bayer Leverkusen“ oder „Borussia M’gladbach“ den entsprechenden Entitäten `smm:Team_Bayer_Leverkusen` bzw. `smm:Team_Borussia_Moenchengladbach` in unserer Datenbank zuzuordnen. Dabei werden zunächst alle Eingabezeichenketten normalisiert, d.h. Akzente und Umlaute entfernt.
Die Zuordnung der Mannschaften geschieht folgendermaßen:

1. Ermitteln der Namen aller Mannschaften aus unserer Datenbank mittels einer SPARQL-Anfrage

2. Vergleiche extrahierte Zeichenkette mit allen Mannschaften mit Hilfe der Levenshtein-Distanz

3. Wähle die Entität mit der geringsten Levenshtein Distanz

Nachdem man die beiden Mannschaften ermittelt hat, kann man ihnen einen eindeutigen Hashtag zuordnen. Am Ende eines Ergebnis-Tweets steht jeweils ein Hashtag, der aus einer Raute mit anschließenden sechs alphanumerischen Zeichen besteht. Die ersten drei Zeichen stellen dabei den Hashtag der ersten Mannschaft, die letzten drei Zeichen den Hashtag der zweiten Mannschaft dar. Dieser wird für zukünftige Anwendungen in der Datenbank gespeichert.
Es verbleibt die Zuordnung des Spielers. Der Twitter-Feed bietet hierbei nur die Nachnamen an. Bei Uneindeutigkeiten wird der erste Buchstabe des Vornamens in der Form „V. Nachname“ angegeben. Im Folgenden wird die Zuordnung der Spieler beschrieben:

1. Ermittle alle Spieler, die in den letzten beiden Jahren eine Begegnung für eine der beiden teilnehmenden Mannschaften gespielt haben mittels einer SPARQL-Anfrage

2. Prüfe zunächst, ob ein Spieler exakt dem zuzuordnenden Spielernamen entspricht: Dies fängt Spielernamen ab, die nur aus einem Künstlernamen bestehen wie „Dan-

te“ oder „Diego“. Bei allen anderen Namen ist diese Prüfung nicht erfolgreich, da bei Twitter nur der Nachname gegeben ist.

3. Wenn kein Treffer, prüfe ob ein Spielername den gesuchten Spielernamen enthält
4. Genau ein Treffer: Treffer wird zugeordnet
5. Mehrere Treffer: Prüfe, ob Vorname im Tweet gegeben ist und filtere nach dem Vornamen
6. Anderenfalls: Keine Zuordnung

Nach Spielende werden die Daten mit den Daten von Fussballdaten.de angereichert (siehe Kapitel **??**). Dabei ist kein weiteres Zuordnen zu unserer Datenbank nötig, da bei den Twitter-Ergebnissen die URIs genau entsprechend dem Benennungsschema von Fussballdaten.de erzeugt werden. Dadurch reichern die Fussballdaten.de-Daten die Twitter-Daten automatisch mit Informationen an, die der Twitter-Feed nicht preisgibt, wie zum Beispiel gelbe Karten, Auswechslungen und beteiligte Spieler. Die doppelt erzeugten Tripel werden in der Datenbank nur einmal gespeichert und stellen somit kein Problem dar.

3.6.3 Parsen von Freitext-Tweets

Beim Parsen von Freitext-Tweets von Fussballdaten.de soll erkannt werden, von welcher Mannschaft ein Tweet handelt, um aktuelle Informationen über eine Mannschaft zu extrahieren. Die Grundidee ist, dass sobald bestimmte, spezifische Mannschaftskennungen in einem Tweet enthalten sind, eine Zuordnung zwischen Mannschaft und Tweet angenommen wird. Diese möglichen Mannschaftskennungen werden aus verschiedenen Datenquellen erzeugt:

1. Splitten des von Fussballdaten.de bekannten Mannschaftsnamen anhand von Leerzeichen: Für Bayer Leverkusen gibt dies zum Beispiel die Token „FC“, „Bayer“, „04“ und „Leverkusen“. Davon werden „FC“ und „04“ durch Blacklisting ausgeschlossen.
2. Ermitteln möglicher Spitznamen aus der DBpedia. Unter dem Prädikat `dbprop:nickname` mit Bayer Leverkusen als Subjekt findet sich zum Beispiel der Spitznamen „Werkself“.
3. Nutzen der Hashtags von früher erkannten Ergebnis-Tweets: Mit Hilfe der bei den Ergebnis-Tweets ermittelten Hashtags kann überprüft werden, ob der Hashtag einer Mannschaft in einem Freitext-Tweet vorkommt.

Diese Mannschaftskennungen werden in einem vorgelagerten Schritt (siehe Schritt 4 aus Kapitel **??**) erzeugt, sodass alle Mannschaftskennungen bekannt sind, wenn zu einem spezifischen Tweet die Mannschaft erkannt werden sollen.
Mit der ermittelten Menge an spezifischen Mannschaftskennungen für eine Mannschaft kann man überprüfen, ob ein gegebener Tweet dieser Mannschaft zugeordnet werden kann, wobei ein Tweet auch mehreren Mannschaften zugeordnet werden kann. Dafür

wird für alle Mannschaftskennungen überprüft, ob sie im Text des Tweets vorkommen. Eine simple Prüfung auf Enthaltensein genügt allerdings nicht, da dabei Tweets über „Bayern“ auch „Bayer Leverkusen“ zugeordnet werden würden. Deswegen sind keine weiteren Buchstaben nach der Kennung erlaubt, außer „s“ und „er“. Dadurch erkennt man gebeugte, häufig vorkommende Wendungen wie „Bayerns Spieler verletzt“ oder „Hamburger Spielmacher wechselt Verein“.
Für Tweets, denen erfolgreich Mannschaften zugeordnet werden konnten, wird eine Verbindung zwischen Tweet und Mannschaft in der Datenbank abgespeichert, um zum Beispiel mit einer Anfrage die fünf letzten Tweets über eine Mannschaft ermitteln zu können.

3.7 Scheduler

Der Scheduler lädt zu definierten Zeitpunkten aktuelle Daten von den Daten-Webseiten herunter, parst die Daten, führt falls notwendig die Zuordnungsalgorithmen aus und speichert die Daten in der Datenbank. Ziel des Schedulers ist es, zu jedem Zeitpunkt aktuelle Daten auf der Webseite präsentieren zu können, gleichzeitig aber die abzurufenden Seiten nicht zu überlasten und in Gefahr zu geraten, wegen zu vieler Anfragen gesperrt zu werden. Dafür sind dem Scheduler alle Spiele mit genauem Datum und Spielbeginn bekannt. Somit müssen nur zu Zeiten, an denen auch wirklich Spiele stattfinden, aktuelle Daten abgefragt werden. Die Twitter-API wird beginnend mit dem Start einer Begegnung im 30-Sekunden-Takt abgefragt. Pro Stunde ergibt das 120 Abfragen, was unter dem von Twitter vorgegeben Maximum von 150 Abfragen pro Stunde liegt. Gleichzeitig ermöglichen Abfragen aller 30 Sekunden einen stets aktuellen Datensatz. Bei jeder Abfrage werden alle Tweets geparst, zugeordnet und die neu erzeugten Tripel werden in der Datenbank gespeichert.
Zusätzlich zur Twitter-API werden auch die Fussballdaten.de-Daten regelmäßig abgefragt. Dies geschieht einmal pro Spieltag um 0.00 Uhr. Zu diesem Zeitpunkt sind neue Daten auf Fussballdaten.de immer bereits eingespeist.

4 Evaluation der Zuordnungsalgorithmen

Zur Evaluation der verschiedenen, in Kapitel **??** erläuterten, Zuordnungsalgorithmen, wurden Statistiken erzeugt, die aufzeigen, wie erfolgreich die jeweiligen Algorithmen sind. Dazu wurde überprüft, wie viel Prozent der jeweiligen Entitäten zugeordnet werden konnten und anhand einer Stichprobe kontrolliert, ob die zugeordneten Entitäten auch korrekt zugeordnet wurden. Die Erklärung der jeweils fehlgeschlagenen Zuordnungen ist auch gegeben.

4.1 Evaluation der Zuordnung von Uefa

Um die Qualität des Algorithmus zu überprüfen, wurden Analysedurchläufe des Programms durchgeführt, deren Ausgaben statistisch ausgewertet wurden.
Insgesamt gibt es 22417 Uefa.com-Spieler, welche zu unserer Basisquelle zugeordnet werden mussten. Davon wurden von dem oben beschriebenem Algorithmus 18116 Uefa.com-

Spieler jeweils ein Fussballdaten.de-Spieler zugeordnet, während 4301 Spieler nicht zugeteilt werden konnten. Dies entspricht einer Trefferquote von 80,81%. Hauptgrund der nicht zuordenbaren Spieler ist das Fehlen des Spielers in der Basisquelle. Weiterhin werden Spieler nicht zugeordnet, deren Namen stark in der Schreibweise variieren. Dies ist zum Beispiel der Fall, wenn die Namen unterschiedlich von kyrillischen Zeichen in die lateinische Schrift transliteriert wurden. Ist die Levenshtein-Distanz infolgedessen größer als der verwendete Schwellwert von drei, wird der Spieler nicht zugeordnet.
Um festzustellen, ob die zugeordneten Spieler tatsächlich korrekt zugeordnet wurden, wurde eine Stichprobe von 400 zugeordneten Spielern zufällig ausgewählt und analysiert. Dabei konnten 372 Zuordnungen als korrekt bestätigt werden. Dementsprechend wird in dieser Stichprobe eine Genauigkeit von 93,00% erzielt und die Anzahl der zugeordneten Spieler in der Gesamtsumme liegt mit einer Wahrscheinlichkeit von 95% zwischen 90,07% und 95,11%. Die Auswirkungen der unterschiedlichen Komponenten des oben beschriebenen Algorithmus auf die Trefferquote wird in Abbildung **??** aufgezeigt.
Fehlerhafte Zuordnungen entstehen durch die Annahme, dass Geburtsdaten der Spieler in den Basisdaten korrekt sind. Diese Annahme wird, wie in **??** bereits beschrieben, aufgrund der falschen Geburtsdaten einiger Spieler in der Uefa.com-Datenquelle getroffen. Wird zu dem Namen eines Uefa.com-Spielers nur ein Pendant in den Basisdaten gefunden, wird von einem inkorrektem Geburtsdatum des Uefa.com-Spielers ausgegangen und der Spieler wird, durch übernehmen des Geburtsdatums des gefundenen Spielers, zugeordnet. Dadurch kann es zu falschen Zuordnungen kommen, wenn der so zugeordnete Spieler nicht dem eigentlichen Spieler entspricht.
Das vorgestellte Vorgehen könnte man noch verbessern, indem man ein Vorgehen entwickelt, bei dem zusätzlich die Vereinsgeschichte des Spielers berücksichtigt und verglichen wird. Dies könnte zu noch besseren Ergebnissen beim Zuordnen führen.

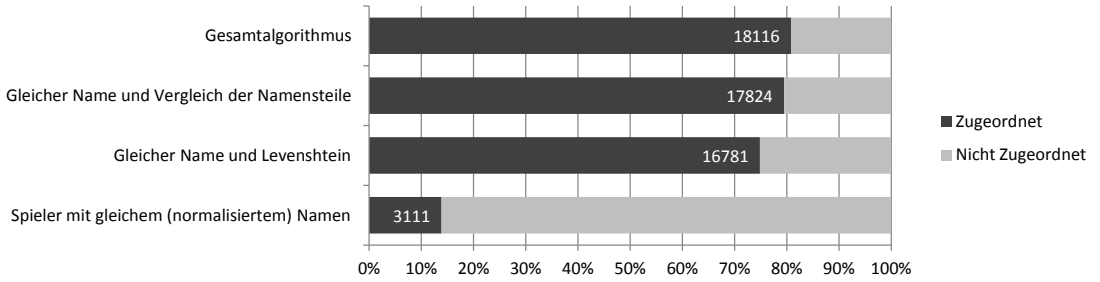

Abbildung 5: Zugeordnete Spieler in Abhängigkeit des verwendeten Algorithmus

Beim Zuordnen von Mannschaften ist zu erwähnen, dass nur deutsche Vereinsmannschaften beachtet werden, da in der Basisquelle nur Mannschaften von deutschen Vereinen enthalten sind. Insgesamt gibt es 25 Mannschaften, welche bei Uefa.com an Begegnungen teilgenommen haben. Der Zuordnungsalgorithmus ordnet dabei 22 Mannschaften zu, wobei die Korrektheit aller Zuordnungen bestätigt werden konnte. Damit kann eine Trefferquote von 88,00% mit einer Genauigkeit von 100,00% verzeichnet werden.
Bei zwei Mannschaften, für die keine Zuordnung gefunden werden konnte, lag dies an einem Namenswechsel im Verlauf der Vereinsgeschichte. So hieß der „FC Erzgebirge Aue“

zwischenzeitlich „SC Wismut Karl-Marx-Stadt“. Diese beiden Mannschaften wurden infolgedessen per Hand zugeordnet.
Die übrige, nicht zuordenbare Mannschaft ist nicht in der Basisquelle enthalten und wird somit als neue Mannschaft in den Triple-Store geladen.

4.2 Evaluation der Zuordnung von DBpedia-Entitäten

Bei der Auswertung des Algorithmus für die Zuordnung von DBpedia-Entitäten auf unsere Entitäten wird besonders auf den Zuordnungsalgorithmus der Spieler und Mannschaften eingegangen, da diese im Kapitel **??** als bedeutendste Algorithmen vorgestellt wurden.

4.2.1 Evaluation der Zuordnung der Spieler

Die Spieler werden in zwei Kategorien unterteilt:

1. Spieler der 1. Bundesliga, die in der Zeit von 1963/1964 bis 1999/2000 gespielt haben.

2. Spieler, die ab der Saison 2000/2001 bis zur Saison 2011/2012 in der 1. oder 2. Bundesliga gespielt haben.

In die erste Kategorie fallen 3435 Spieler. Von diesen konnten 2438 einer oder mehreren DBpedia-Entitäten zugeordnet werden, dies entspricht 70,98%. Es konnten also 29,02% (997 Spieler) nicht zugeordnet werden. In der zweiten Kategorie konnten von 3355 Spielern 2865 Spieler zugeordnet werden (85,39%). Bei 490 (14,61%) Spielern schlug die Zuordnung fehl.
Insgesamt konnten also von 6790 Spielern 5303 Spieler zugeordnet werden (78,10%). Um die Korrektheit des Ergebnisses zu prüfen, wurden in beiden Kategorien jeweils 300 zugeordnete Spieler zufällig ausgewählt und analysiert. Alle 600 Spieler wurde richtig zugeordnet. Damit beträgt die Wahrscheinlichkeit der korrekt zugeordneten Spieler in der Gesamtsumme mit einer Wahrscheinlichkeit von 95% über 98,74%.
Die Gründe, weshalb manche Spieler nicht zugeordnet werden konnten, sind in beiden oben genannten Fällen gleich. Im Statistik-Anhang befindet sich eine Analyse einer Stichprobe von 50 nicht zugeordneten Spielern. Auf diese Stichprobe beziehen sich die nun folgenden Statistiken.
Einer der beiden Hauptgründe, warum Spieler nicht zugeordnet werden konnten, ist, dass es zu manchen Spielern keine DBpedia-Seite gibt, sodass gar keine Zuordnung möglich ist. Dies war in der ersten Kategorie zu 52% und in der zweiten Kategorie zu 42% bei der Stichprobe der Fall.
Der andere Hauptgrund ist, dass sich die Namen der beiden Entitäten nicht genug ähneln, sodass sie das Levenshtein-Kriterium nicht erfüllen. Bei der Stichprobe traf dies im ersten Fall bei 18 Spielern zu (36%) und im zweiten Fall bei 25 Spielern (50%). Dies kann daran liegen, dass wir einen anderen Namen der Person gespeichert haben als DBpedia. So wird zum Beispiel der Spieler „Joseph Enochs“ (voller Name) nicht gefunden, da dieser Spieler in der DBpedia unter seinem Spitznamen „Joe Enochs“ vermerkt ist. Eine zu stark abweichende Schreibweise der Namen stellt auch ein Problem

bei der Zuordnung dar. Dieser Fall tritt auf, wenn in der DBpedia der Spielername mit landestypischen Sonderzeichen aufgeführt ist, im Triple-Store jedoch der normalisierte Name eingetragen ist, sodass die Namen sich zu sehr unterscheiden. Außerdem stellt das Encoding des deutschen DBpedia Endpunktes bei dem Namensvergleich ein Problem dar. Aus unerklärlichen Gründen werden Zeichenketten geliefert, die jenseits normaler ASCII-Zeichen nicht mehr interpretierbar sind. Laut eines Mitarbeiters der DBpedia sollten diese Zeichen Unicode-escaped geliefert werden, dies ist jedoch nicht der Fall[8]. Durch die fehlerhafte Kodierung unterscheiden sich die Namen zu stark, sodass die vermutlich richtige DBpedia-Seite verworfen wird.
Ein letzter Grund, warum Spieler nicht zugeordnet werden können, ist, dass der Vergleich des Geburtsdatums fehl schlägt. Dies tritt auf, wenn ein falsches Geburtsdatum in einer der beiden Quellen gespeichert ist. Dies ist bei der Stichprobe in der ersten Kategorie zu 12% und in der zweiten Kategorie zu 8% der Fall.
Um den Zuordnungsalgorithmus von Spielern noch zu verbessern, gilt es das Encoding-Problem zu lösen. Des Weiteren würde eine Verfeinerung der Normalisierung der Namen Verbesserungen mit sich bringen, da Sonderzeichen im Namen der Spieler-Entitäten eine Fehlerursache darstellen.

4.2.2 Evaluation der Zuordnung der Mannschaften

Bei der Evaluation der Mannschaften wurden nur Mannschaften betrachtet, die mindestens einmal in der 1. oder 2. Bundesliga gespielt haben. Der Triple-Store enthält insgesamt 54 dieser Mannschaften. Von diesen konnten 100% einer DBpedia-Entität zugeordnet werden. Zum Überprüfen der Korrektheit der Zuordnung wurden alle Mannschaften überprüft: Alle wurden korrekt zugeordnet.

4.2.3 Evaluation der Zuordnung weitere Entitäten

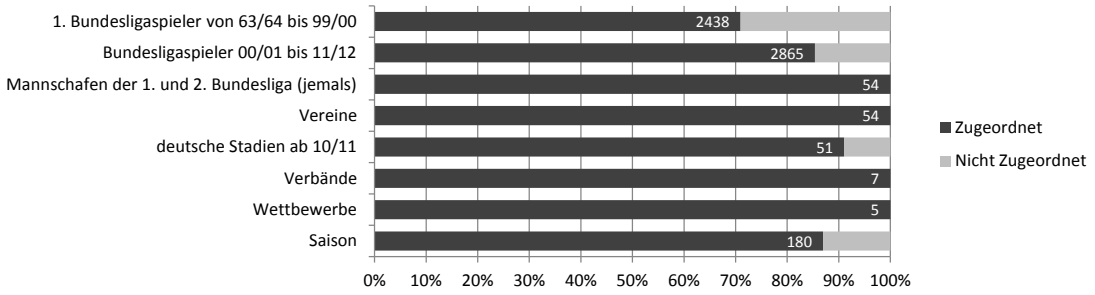

Abbildung 6: Trefferquote von allen auf die DBpedia zugeordneten Entitäten

In Abbildung **??** ist die Trefferquote der einzelnen Entitäten dargestellt. Neben den oben beschrieben Fällen sind auch die Statistiken zu den Wettbewerben, Verbänden, Vereinen, Saisons und Stadien dargestellt.

[8] `http://www.mail-archive.com/dbpedia-discussion@lists.sourceforge.net/msg02711.html`

Die Wettbewerbe und Verbände haben eine Trefferquote von 100%, da die wenigen Entitäten per Hand zugeordnet werden. Auch die 54 Vereine wurden alle korrekt zugeordnet.

Bei den Saisons wurden 180 von 210 und damit 86,96% zugeordnet. Die zugeordneten Saisons sind alle Saisons der 1. oder 2. Bundesliga, der Champions League, der Weltmeisterschaften und Europameisterschaften. Alle nicht gefundenen Saisons gehören zur Bundesliga Süd, zur Bundesliga Nord oder sind Relegationen, zu denen es keine DBpedia-Seite gibt, sodass diese auch nicht zugeordnet werden können. Alle zugeordneten Saisons sind richtig zugeordnet.

Die Evaluation der Stadien begrenzt sich auf solche, die Austragungsort einer Begegnung einer Saisons ab 2010/2011 der 1. und 2. Bundesliga waren. Dabei handelt es sich um 56 Stadien. Diese Begrenzung wurde vorgenommen, da sich die Namen der Stadien aufgrund der häufigen Sponsorenwechsel oft ändern.
Von den 56 betrachteten Stadien konnten 51 einer Entität aus der DBpedia zugeordnet werden, was 91,07% entspricht.
Fünf Stadien konnten also nicht zugeordnet werden. Zu dem Stadion „Benteler-Arena“ in Paderborn gibt es keine DBpedia-Seite, sodass dies nicht gefunden werden konnte. Zum „Jahn-Stadion“ wurde keine passende DBpedia-Entität gefunden, da in der DBpedia der Name „Jahnstadion“ eingetragen ist und somit die Überprüfung auf Enthaltensein des Namens in der SPARQL-Anfrage fehlschlägt. Bei den übrigen drei Stadien sind in unserem Virtuoso Triple-Store inkonsistente, veraltete Namen gespeichert (z. B. „Bremer Brücke“), sodass unser Name dem aktuellen Namen aus der DBpeida nicht genügend ähnelt und keine DBpedia-Entität gefunden werden kann.
Von den 51 zugeordneten Stadien wurden drei falsch zugeordnet. Bei zwei Stadien wurden neben den richten DBpedia-Entitäten noch ein weiteres falschen Stadion zugeordnet, welches den gleichen Namen trägt, jedoch in einer anderen Stadt steht (z. B. wurde „Weserstadion“ in Bremen dem „Weserstadion“ in Minden zugeordnet). Bei einem Stadion wurde die DBpedia-Entität einer Mühle zugeordnet, nach der das Stadion benannt wurde („Lohmühle in Lübeck“).
Eine Verbesserung der Ergebnisse wäre durch die Beachtung des Standorts des Stadions möglich. Gleichnamige Stadien würden so dem Stadion zugeordnet werden, in der das Stadion steht. Eine Überprüfung der DBpedia-Entität um sicherzustellen, dass es sich um ein Gebäude bzw. Stadion handelt, würde ebenfalls Fehlerquellen eliminieren.

4.3 Evaluation der Twitter-Ergebnisse

Von insgesamt 2127 Ergebnis-Tweets werden 2022 korrekt zugeordnet, bei 105 schlägt die Zuordnung fehl.
Von diesen 105 schlägt bei 97 Ergebnis-Tweets die Zuordnung fehl, da es sich nicht um Bundesliga-Spiele handelt sondern um Länderspiele. Hier besteht die Vermutung, dass diese Tweets fälschlicherweise unter dem falschen Benutzerkonto veröffentlicht wurden. Da bei der Zuordnung der Mannschaften nur unter allen bisherigen Bundesliga-Mannschaften gesucht wird, wird keine Mannschaft gefunden, deren Levenshtein-Distanz zu den geparsten Mannschaften klein genug ist, und die Zuordnung schlägt fehl. Diese

Fehler werden ignoriert, da in diesem Fall ausschließlich die Zuordnung von Bundesliga-Mannschaften relevant ist.
Weiterhin gibt es acht Tweets, die nicht zugeordnet werden können, weil zu viele mögliche Spieler gefunden werden. In sieben Fällen scheitert die Zuordnung, weil Spieler doppelt in der Datenbank vorhanden sind. Dementsprechend werden zwei Spieler gefunden. Diese sieben Fälle werden von nur zwei Spielern verursacht. In einem Fall kam der Name des Torschützens („Müller“) dreimal in einer Partie vor, es wurde allerdings kein Vorname angegeben.
Zieht man die irrelevanten Tweets ab, ist also in 2022 von 2030 Fällen eine Zuordnung möglich, was eine Trefferquote von 99,6 % ergibt. Bei einer Stichprobe von 200 zufällig ausgewählten Tweets waren alle Zuordnungen korrekt, die Genauigkeit liegt also mit 95 %-iger Wahrscheinlichkeit über 98.12 %.
Bei den Texttweets werden von insgesamt 1780 Tweets 1574 einem Team zugeordnet. In allen Fällen ohne Zuordnung war kein Mannschaftsname genannt, sodass keine Erkennung möglich war. Eine Stichprobe (siehe Statistik-Anhang) der Größe 150 ergab auch hier, dass alle Zuordnungen korrekt waren (mit 95 %-iger Wahrscheinlichkeit über 97,5 %). Abbildung **??** zeigt die Ergebnisse zusammengefasst.

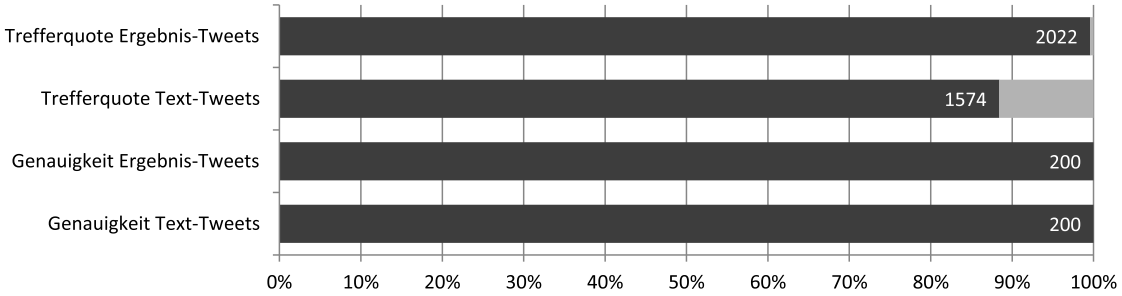

Abbildung 7: Ergebnisse der Zuordnung von Twitter-Ergebnis-Tweets und Freitext-Tweets

Eine mögliche Verbesserung wäre, nicht nur auf die Mannschaftsnamen zu achten, sondern auch auf Trainer- und Spielernamen. Damit könnte man noch mehr der Freitext-Tweets zuordnen, würde aber wahrscheinlich einen Teil der Genauigkeit verlieren, da Spielernamen oft doppelt vorkommen und in der Schreibweise variieren.

5 Zusammenfassung und Ausblick

In dieser Arbeit haben wir Verfahren vorgestellt, um aus diversen Internet-Datenquellen einen möglichst konsistenten, vollständigen und korrekten Fußballdatensatz zu erstellen. Wir haben eine Basisdatenquelle festgelegt, welche bereits eine Vielfalt von historischen Fußballdaten enthält. Diese wurde mit Daten anderer, sowohl live als auch nicht-live, Quellen angereichert. Bei dieser Anreicherung wurde besonders auf die Konsistenz und

Korrektheit der zusammengeführten Daten geachtet. Der somit erstellte Datensatz wurde in Form einer Webseite visualisiert und benutzerfreundlich zur Verfügung gestellt.
Ein möglicher Anknüpfungspunkt an diese Arbeit wäre das Einbeziehen von mehr Datenquellen. Insbesondere Datenquellen, die besondere Informationen zu einem Spieler in einer Begegnung enthalten, wie zum Beispiel die Anzahl der gelaufenen Kilometer, die Anzahl der Pässe, die Fehlpassquote oder die Anzahl der Torschüsse. Auch könnte man den Datenumfang noch erweitern, indem man Datenquellen hinzuzieht, die Informationen über mehr Ligen beinhalten, wie zum Beispiel die dritte deutsche Liga. Das Parsen von Vereinsseiten zum Gewinnen von Informationen über die Vereingeschichte, den Mitgliedsschaftsantrag und den Ticketshop stellt eine weitere Möglichkeit für neue Datenquellen dar.
Ein anderer möglicher Anknüpfungspunkt wäre es, komplett natürlichsprachliche Texte zu verarbeiten. Durch Techniken des Natural Language Processing könnte man aus Berichten von Fachzeitschriften Informationen über Wechsel, Verletzungen und Interviewaussagen gewinnen.

A Glossar

Begegnung: Begegnung bezeichnet ein Spiel zweier Mannschaften.

DBpedia: Ein Projekt, das Daten von Wikipedia strukturiert zur Verfügung stellt.

Entität: Eindeutig identifizierbare Teile der Domäne, wie z. B. ein Spieler, eine Begegnung oder ein Team.

Genauigkeit: Anzahl der richtig zugeordneten Entitäten dividiert durch die Gesamtanzahl der Entitäten einer Stichprobe, auch bekannt als Precision.

JSON: Datenaustauschformat mit Schlüssel-Wert-Paaren.

Levenshtein: Ähnlichkeit zweier Zeichenketten, gemessen anhand der benötigten Ersetzungen, Löschungen und Einfügungen von Zeichen, um die Zeichenketten anzugleichen.

Named Entity Recognition: Erkennen von Entitäten in einem freiem Text.

Parsen: Verarbeiten einer beliebigen Eingabe (z.B. JSON- oder HTML-Datei) in eines für die Weiterverarbeitung günstigen Formats.

RDF: Abkürzung für Resource Description Framework, gerichtetes beschriftetes Graphdatenformat [**?**].

Regulärer Ausdruck: syntaktische Beschreibung von Zeichenketten.

Scheduler: Programmeinheit, welche zu definierten Zeitpunkten die Daten neu parst.

SPARQL: Abfragesprache für RDF.

Trefferquote: Anzahl der zugeordneten Entitäten dividiert durch die Gesamtanzahl der Entitäten, auch bekannt als Recall.

Tweet: Tweet Kurznachricht (maximal 140 Zeichen) eines Nutzers auf `twitter.com`.

Twitter-Feed: Chronologische Liste aller Tweets eines Nutzers auf `twitter.com`.

URI: Abkürzung von Uniform Resource Identifier, eindeutig identifizierbarer Bezeichner einer Ressource.

Virtuoso Triple-Store: Datenbank, in welcher RDF-Tripel gespeichert werden können.

XML: Abkürzung für Extensible Markup Language, strukturiertes menschenlesbares Datenformat.

Zuordnen: Finden des Pendants einer Entität in unserer Datenbank.

B Präfixe

Präfixe, die in SPARQL-Anfragen verwendet werden.

xsd	http://www.w3.org/2001/XMLSchema#
dbprop	http://DBpedia.org/property/
owl	http://www.w3.org/2002/07/owl#
rdfs	http://www.w3.org/2000/01/rdf-schema#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
time	http://www.w3.org/2006/time#
smm	http://purl.org/hpi/soccer-voc/
foaf	http://xmlns.com/foaf/0.1/
dbpo	http://DBpedia.org/ontology/

C Beispielanfragen

C.1 Spieler mit den meisten Eigentoren

Listing 5: SPARQL-Anfrage zur Ermittlung des Spielers mit den meisten Eigentoren

```
PREFIX smm: <http://purl.org/hpi/soccer-voc/>
SELECT (COUNT(?goal) AS ?goals) ?scorer
FROM <http://hpi.uni-potsdam.de/smm2013group3>
WHERE
{
  ?team <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> smm:Team.
  ?match smm:partOf ?season.
  ?season smm:associatedWith smm:Competition_1__Bundesliga.
  ?goal smm:match ?match.
  ?goal smm:forTeam ?team.
  ?goal <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> smm:Goal.
  ?goal smm:scorer ?scorer.
  ?goal smm:ownGoal 1.
}
GROUP BY ?scorer
ORDER BY DESC(?goals)
LIMIT 3
```

C.2 Strengster Schiedsrichter

Listing 6: SPARQL-Anfrage zur Ermittlung des Schiedsrichters mit den meisten Karten pro Spiel

```
PREFIX smm: <http://purl.org/hpi/soccer-voc/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT DISTINCT ((xsd:float(COUNT(?card)) / ?reffed) as ?cardspergame)
                ?name
                ?referee
```

```
FROM <http://hpi.uni-potsdam.de/smm2013group3>
WHERE
{
    ?card smm:givenByReferee ?referee .
    ?referee smm:name ?name .
    {
        SELECT DISTINCT (COUNT(?referee) as ?reffed) ?name
        FROM <graph>
        WHERE
        {
            ?match rdf:type smm:Match .
            ?match smm:referee ?referee .
            ?referee smm:name ?name
        }
        GROUP BY ?reffed ?name
        ORDER BY DESC(?reffed)
    }
    FILTER (?reffed > 10)
}
GROUP BY ?referee ?reffed ?name
ORDER BY DESC(?cardspergame)
LIMIT 10
```

C.3 Ewige Tabelle

Listing 7: Ewige Tabelle der Bundesliga

```
prefix dbprop: <http://dbpedia.org/property/>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix time: <http://www.w3.org/2006/time#>
prefix smm: <http://purl.org/hpi/soccer-voc/>

SELECT ?team
       (MAX(?team_name) as ?team_name)
       (MAX(?team_logo) as ?team_logo)
       (COUNT(?victory) * 3 + COUNT(?remis) as ?points)
       (COUNT(?victory) as ?victories)
       (COUNT(?remis) as ?draws)
       (COUNT(?defeat) as ?defeats)
       (SUM(?goal) as ?team_goal)
       (SUM(?other_goal) as ?team_other_goal)
FROM <http://hpi.uni-potsdam.de/smm2013group3>
WHERE
{
  ?team smm:name ?team_name .
  ?team smm:teamLogo ?team_logo .

  {
    ?victory smm:partOf %1$s.
    ?victory smm:homeTeam ?team .
    ?victory smm:finalHomeScore ?goal .
    ?victory smm:finalGuestScore ?other_goal .
    FILTER(?goal > ?other_goal)
```

```
  }
  UNION
  {
    ?victory smm:partOf %1$s .
    ?victory smm:guestTeam ?team .
    ?victory smm:finalHomeScore ?other_goal .
    ?victory smm:finalGuestScore ?goal .
    FILTER(?other_goal < ?goal)
  }

  UNION
  {
    ?remis smm:partOf %1$s.
    ?remis smm:homeTeam ?team .
    ?remis smm:finalHomeScore ?goal .
    ?remis smm:finalGuestScore ?other_goal .
    FILTER(?goal = ?other_goal)
  }
  UNION
  {
    ?remis smm:partOf %1$s .
    ?remis smm:guestTeam ?team .
    ?remis smm:finalHomeScore ?other_goal .
    ?remis smm:finalGuestScore ?goal .
    FILTER(?other_goal = ?goal)
  }

  UNION
  {
    ?defeat smm:partOf %1$s .
    ?defeat smm:homeTeam ?team .
    ?defeat smm:finalHomeScore ?goal .
    ?defeat smm:finalGuestScore ?other_goal .
    FILTER(?goal < ?other_goal)
  }
  UNION
  {
    ?defeat smm:partOf %1$s .
    ?defeat smm:guestTeam ?team .
    ?defeat smm:finalHomeScore ?other_goal .
    ?defeat smm:finalGuestScore ?goal .
    FILTER(?other_goal > ?goal)
  }
}
GROUP BY(?team)
ORDER BY DESC(?points)
```