Fachhochschule Brandenburg
Fachbereich Informatik und Medien

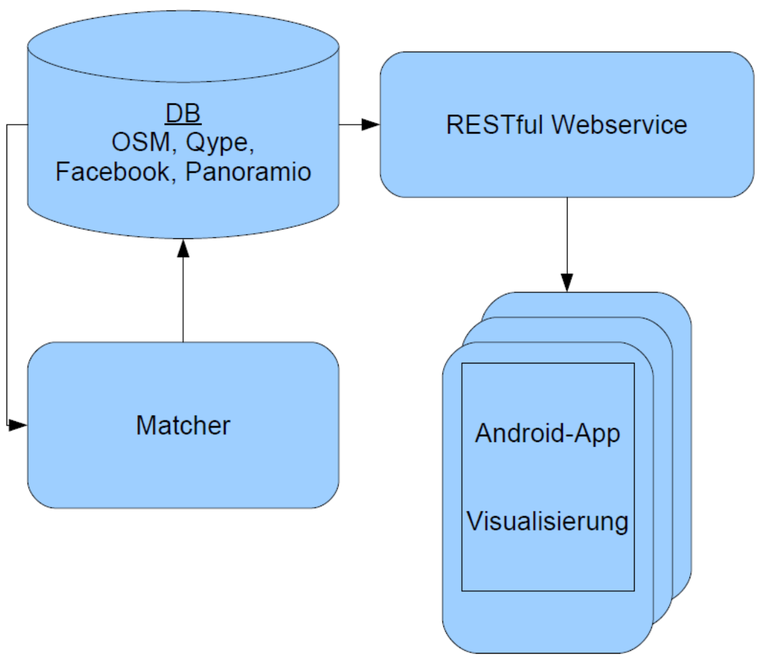
Matching und Visualisierung von Points of Interest aus sozialen Daten

Bachelorarbeit, vorgelegt von Paul Lehmann

Aufgabenstellung

Ziel dieser Arbeit ist es, Daten zu Points of Interest (PoI), wie Restaurants, Bars, Sehenswürdigkeiten, aus verschiedenen Netzwerken zusammenzulegen, zu aggregieren. Dieses Problem ergibt sich, da die Metadaten der PoIs in den verschiedenen Netzwerken jeweils unterschiedlich repräsentiert sind, und auch fehlerhafte Daten enthalten sein können. Dabei sollen mehrere Verfahren überprüft und das mit dem besten Ergebnis gefunden werden. Die Netzwerke, deren Daten verwendet wurden, sind: Qype, Facebook Places und Panoramio. Ihre PoIs sollen jeweils auf Punkte aus OpenStreetMap (OSM) gematcht werden. Die aggregierten Daten müssen auf einer Karte in einer Android-Anwendung visualisiert werden.

Systemarchitektur



In der Datenbank waren bereits vorher die Daten vorhanden. Mit diesen Daten werden die Matches gesucht, anschließend werden die Matches wieder in die Datenbank eingetragen und vom Webservice bei Nachfrage an die App weitergeleitet.

Algorithmen

Vorraussetzung: Als Attribute sind nur Koordinaten und Titel für die PoIs in allen Netzwerken durchgängig vorhanden.

Gewählte Algorithmen:

Longest common substring

Gesucht ist hierbei der längste gemeinsame Teilstring von 2 oder mehr Zeichenketten. Dieses Verfahren ist schnell und einfach implementiert. Außerdem liefert es eine hohe Anzahl an richtigen Ergebnissen, allerdings auch eine hohe Anzahl an falschen Ergebnissen.

Nähester Point of Interest

Bei diesem Verfahren wird der PoI als Match zugewiesen, der den geringsten Abstand hat. Berechnet wird diese Distanz mithilfe des Euklidischen Abstandes. Dieses Verfahren ist ebenfalls schnell und einfach implementiert. Allerdings erreicht es wenige richtige und viele falsche Matches.

Geofilter und Stringvorverarbeitung

Zum einen filtert der Geofilter die komplette Liste der OSM-PoIs. Als Kandidaten werden die ausgewählt, die in einer Bounding Box um den Punkt herum liegen. Die Stringvorverarbeitung wandelt die Sonderzeichen der deutschen Sprache (ä,ö,ü,ß) in (ae,oe,ue,ss) um. Anschließend werden alle nichtalphanumerischen Zeichen durch Leerzeichen ersetzt. Mehrere Leerzeichen in Folge werden zu einem reduziert.

Edit-Distance (Levenshtein-Distance)

Dieses Verfahren bestimmt die minimale Anzahl an Operationen, die benötigt werden, um eine Zeichenkette in eine andere zu überführen. Die zulässigen Operationen sind Einfügen, Löschen und die Substitution (Ersetzen). Dieses Verfahren verzeiht Tippfehler und es gibt wenige falsche Matches.

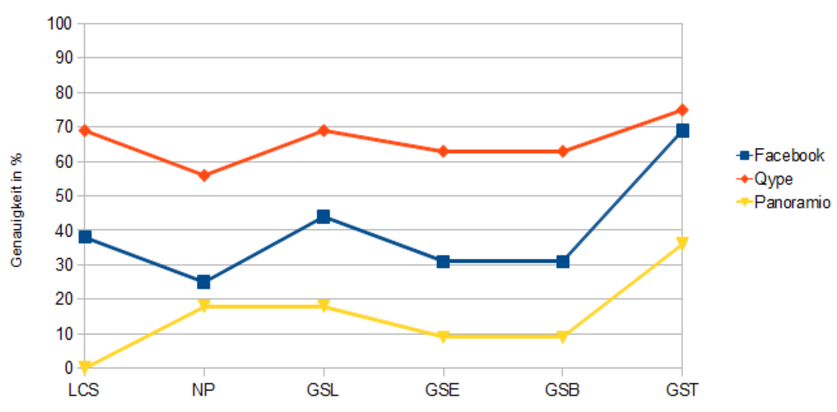
Bigram-Distance

Bei diesem Verfahren teilt man die Zeichenketten in kleinere Blöcke der Länge 2, die sich überschneiden, dabei ist der letzte Buchstabe stets der erste des nächsten Blockes. Die Distanz zwischen den beiden Zeichenketten ist die Anzahl der Blöcke, die sich gleichen. Fehlende oder überflüssige Zeichen haben auf diesem Wege kaum Einfluss. Außerdem gibt es wenige falsche Ergebnisse. Da die Reihenfolge relevant ist, gibt es Probleme bei PoIs, deren Titel in anderer Reihenfolge repräsentiert sind.

TF-IDF-Wichtung

Dieses Verfahren ist im Gegenteil zu den vorherigen term- bzw. wortbasiert. Terme bekommen Wichtungen, die belegen, wie oft ein Term in dem Dokument bzw. in einem Korpus von Dokumenten vorkommt. Je häufiger ein Term in einem Dokument vorkommt, desto mehr fällt er ins Gewicht. Kommt dieser Term allerdings häufig im gesamten Korpus vor, dann wird er weniger gewichtet. Der Vorteil ist, dass die Reihenfolge der Terme keine Rolle spielt. Sehr viele Ergebnisse sind richtig und nur sehr wenige bis gar keine falsch. Da dieses Verfahren allerdings wortbasiert, treten bei kleinsten Abweichungen in den Termen, beispielsweise Tippfehler, Probleme auf.

Ergebnisse



Genauigkeit der verschiedenen Verfahren

In der Abbildung sieht man die Genauigkeiten der Verfahren, die verwendet wurden. Die Abkürzungen stehen dabei für: longest common substring (LCS), nähester PoI (NP), LCS erweitert mit Geofilter und Stringvorverarbeitung (GSL), Edit-Distance (GSE), Bigram-Distance (GSB) und TF-IDF (GST) (die letzten 3 ebenfalls durch Geofilter und Stringvorverarbeitung erweitert)

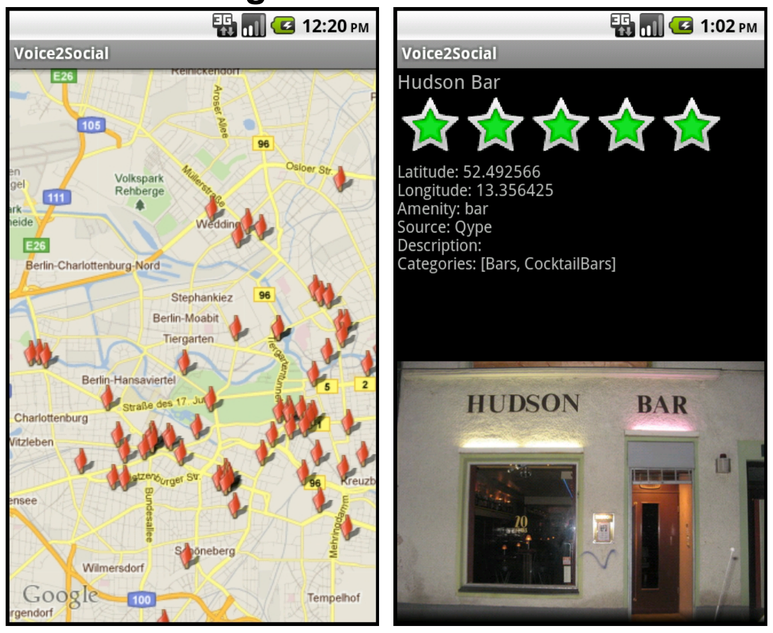
Die drei Graphen zeigen dabei die drei Netzwerke, aus denen die PoIs auf die PoIs auf OSM gematcht wurden. Die Daten von Panoramio haben sich nicht geeignet um das Matching durchzuführen, weil die Titel oft nicht brauchbar sind. Die Daten von Qype und Facebook Places waren dagegen besser geeignet. Zu sehen ist, dass TF-IDF stets die höchste Genauigkeit erzielt hat. Außerdem hatte es auch stets die wenigsten falschen Matches aller Verfahren.

Webservice-Rückgabe (JSON)


```
{
 - facebookMatches: {
    - entry: [
       - {
          - key: {
               latitude: "52.498131626224",
               longitude: "13.35426419481",
               name: ""Cafe Xara"",
               checkin_count: "0",
               description: "",
               id: "140040352709524"
            },
          - value: {
               amenity: "cafe",
               id: "819449789",
               latitude: "52.4980592",
               longitude: "13.3545155",
               name: "xara cafe lounge"
            }
         },
```


Beispiel einer solchen Rückgabe

Visualisierung der Daten



Visualisierung der PoIs (links), PoI-Beispiel (rechts)

Zur Visualisierung der Daten wurde eine Android-Anwendung des DFKI erweitert. Es wurde eine Anbindung an den Webservice, eine Interpretation der Antwort des Webservices und die Darstellung der PoIs als Fahnen (Abb. links) implementiert. Beim Antippen dieser Fahnen bekommt man Informationen zu diesen PoIs (Abb. rechts).

Fazit/Ausblick

Das Verfahren TF-IDF hat von den verwendeten Verfahren das beste Ergebnis erzielt, in dem es die falschen Matches vermieden hat. PoIs, zu denen kein Match gefunden wurde, der ähnlich genug ist, blieben ohne Match. Überraschend war das Ergebnis des *longest common substring,* das sehr viele richtige Ergebnisse geliefert hat.

Als weiteren Schritt im Rahmen des Projektes könnte man den Matchingalgorithmus TF-IDF erweitern, um die potenzielle Fehlerquelle der Tippfehler zu berücksichtigen, beispielsweise mithilfe der Edit-Distance. Man könnte alle Terme als gleich deklarieren, deren Edit-Distance kleiner oder gleich 1 ist.

Betreuer: Dipl.-Inf. I. Boersch (FH Brandenburg)
Dr. T. Scheffler (DFKI-Projektbüro Berlin)

Abgabe: 31.08.2011