Institut für Formale Methoden der Informatik

Abteilung Algorithmik

Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart

Masterarbeit

Natürlichsprachliche Anfragen an OSCAR

Benjamin Kopf

Studiengang: Informatik, M.Sc.

Prüfer: Prof. Dr. Stefan Funke

Betreuer: Prof. Dr. Stefan Funke

begonnen am: 24.07.2019

beendet am: 24.01.2020

Zusammenfassung

Das rapide Wachstum von OpenStreetMap stellt große Mengen geografischer Daten frei zur Verfügung. So können neue kartenbasierte Applikationen entstehen, welche mit diesen Daten arbeiten. Eine solche Applikation ist OSCAR, eine Suchmaschine die auf den Daten von OpenStreetMap basiert. Die komplexe Query Syntax macht den Einstieg in die Nutzung dieses Systems jedoch schwierig. In dieser Arbeit wird ein natürlichsprachlicher Queryansatz für OSCAR entwickelt, der vor allem Nutzern ohne OpenStreetMap Kenntnisse den Einstieg in OSCAR erleichtern soll. Die Sprache wird von einem Automaten gesteuert, dessen Zustand die Worte vorgibt, die in diesem Moment geschrieben werden können. Eine Usability Nutzerstudie zeigt, dass unerfahrene Nutzer mit diesem System effektiver arbeiten können als mit nativem OSCAR und gleichzeitig in der Lage sind, die Vorteile des OSCAR Suchsystems gegenüber der etablierten Kartenplattform Google Maps zu erkennen.

Inhaltsverzeichnis

1 Einleitung

Seit der Gründung im Jahr 2004 entwickelt sich OpenStreetMap[1] (OSM) zu einer immer größeren Quelle frei verfügbarer, geografischer Daten. Dies ermöglicht die Entwicklung von neuen kartenbasierten Anwendungen, welche auf diese Daten zugreifen. Eine solche Anwendung ist die Suchmaschichne OSCAR[2] von Bahrdt und Funke (2015). Diese ermöglicht das Durchsuchen der in OSM getaggten Objekten mit Hilfe von Key-Value-Paaren (OSM Tags) und Locations.

Ein Problem von OSCAR ist, dass es eine sehr komplexe Querysprache besitzt. Diese ermöglicht zwar dem erfahrenen Nutzer das Erstellen von verschachtelten und dadurch präzisen Anfragen, kann aber gerade für neue Benutzer eine Einstiegshürde darstellen. So kann beispielsweise mit der Query *@amenity:fast_food munich - (burger? ?king) - (mc? ?donald?)* nach Fast Food Restaurants in München gesucht werden, die weder McDonald's noch Burger King Restaurants sind. Jedoch müssen dem Nutzer die Tags und die Syntax bekannt sein, um OSCAR nutzen zu können. Andernfalls müssen die Tags während der Nutzung herausgesucht werden. Mit dem Ziel OSCAR für neue Nutzer leichter zugänglich zu machen, wird in dieser Arbeit ein natürlichsprachlicher Ansatz für die OSCAR Suchmaschine entworfen.

1.1 Aufgabenstellung

Das Ziel dieser Arbeit ist die Umsetzung eines möglichst natürlichsprachlichen Ansatzes für OSCAR. Diese Umsetzung ist auf der Webseite OSCAR FGS[3] zu sehen. Durch den natürlichsprachlichen Ansatz soll OSCAR vor allem neuen Nutzern ohne OSM Kenntnisse zugänglicher gemacht werden. Die Sprache wird dabei von einem Automaten vorgegeben. Je nach eingegebenem

[1] `https://www.openstreetmap.org/`
[2] `http://oscar-web.de/`
[3] `https://fgs.oscar-web.de/`

Wort wechselt dieser in einen anderen Zustand. Der aktuelle Zustand gibt vor, welche Worte zu dem jeweiligen Zeitpunkt eingegeben werden können.

Um die Usability von diesem natürlichsprachlichen Ansatz an OSCAR zu bewerten, wird eine Nutzerstudie durchgeführt. In dieser wird der natürlichsprachliche Ansatz mit nativem OSCAR und mit Google Maps[4] verglichen.

1.2 Verwandte Arbeiten

Diese Thesis basiert hauptsächlich auf der Arbeit von Bahrdt und Funke (2015). In dieser Arbeit wird die OSCAR Suchmaschine vorgestellt. Besonders relevant ist dabei die Syntax der OSCAR Query. Das Ziel dieser Thesis ist ein Übersetzer, welcher natürlichsprachliche Queries an OSCAR in reguläre OSCAR Queries übersetzen kann. Weitere Spezifikationen der OSCAR Query finden sich in Bahrdt (2013).

Das für die natürlichsprachlichen Anfragen benötigte Mapping von Aktivitäten auf OSM Tags wird durch die Arbeit von Codescu et al. (2011a) erleichtert. Darin werden zwei Ontologien aufeinander gemappt. Eine ist eine Ontologie für räumlich lokaliserte Aktivitäten. Die andere ist eine zuvor entwickelte Ontologie für OSM Tags mit dem Namen OSMonto (Codescu et al., 2011b). Durch das Mapping dieser beiden Ontologien können verschiedenen räumlich lokalisierten Aktivitäten OSM Tags zugeordnet werden.

1.3 Gliederung

In Kapitel 2 wird die Kartenplattform OSCAR vorgestellt. Dabei wird vor allem die Querysprache genauer analysiert.

Kapitel 3 befasst sich mit einigen Ontologien von Codescu et al. (2011a). Mit Hilfe dieser Ontologien können lokalisierte Aktivitäten direkt auf OSM

[4] `https://www.google.de/maps/`

Tags gemappt werden. Dadurch wird der *was?* Teil der natürlichsprachlichen OSCAR Query abgedeckt.

Die Umsetzung der natürlichsprachlichen Anfragen an OSCAR wird in Kapitel 4 beschrieben. Der Querysprache wird ein deterministischer endlicher Automat (DEA) zu Grunde gelegt. Dieser schlägt je nach aktuellem Zustand andere Worte vor. Welches Wort eingegeben wird, bestimmt in welchen Zustand sich der Automat begibt. Die Umsetzung des Systems sowie eine kritische Bewertung befinden sich ebenfalls in diesem Kapitel.

Das implementierte System wurde durch eine Nutzerstudie geprüft. Diese wird in Kapitel 5 vorgestellt und ausgewertet. Dabei wird mit Hilfe der Ergebnisse evaluiert, in wie weit die natürlichsprachlichen Anfragen an OSCAR die gewünschten Eigenschaften enthalten.

Zuletzt wird die Arbeit in Kapitel 6 zusammengefasst. Abschließend wird ein Ausblick über mögliche Anknüpfungspunkte gegeben, welche für spätere Arbeiten interessant sein könnten.

2 OSCAR

OSCAR (**O**pen**S**treetMap **C**ell **AR**rangement) ist ein Kartendienst basierend auf Daten von dem Geodatenprojekt OSM (Bahrdt und Funke, 2015). Diese OSM-Daten sind mit mehreren Tags in Form von Key-Value-Paaren versehen, nach denen mit der OSCAR Suchmaschine gesucht werden kann. Hinterlegt sind die einzelnen Objekte im JSON (JavaScript Object Notation) Format. Das Stuttgarter Restaurant *Ganesha* sieht dabei zum Beispiel folgendermaßen aus:

"id":47926310

"osmid":2081981047

"type":"node"

"score":20

"bbox":[48.780376,48.780376,9.2025478,9.2025478]

"shape":{"t":-1}

"k":["addr:city", "addr:housenumber", "addr:postcode", "addr:street", "amenity", "cuisine", "name", "name:de", "opening_hours", "phone", "website", "wheelchair"]

"v":["Stuttgart", "19", "70186", "Lembergstraße", "restaurant", "indian", "Ganesha", "Ganesha Restaurant", "Mo-Sa 11:30-14:30, 17:30-23:30; Su 12:00-23:00; PH 12:00-23:00", "+49 711 4687981", "http://www.ganesharestaurant.de", "no"]

"p":[598957,436098,28194,5989,570,64]

Dabei ist links von dem Doppelpunkt immer der JSON Key und rechts davon der dazugehörige JSON Value zu finden. Die eigentlichen OSM Tags findet man dabei in den beiden Arrays mit den Keys "k" und Values "v". Dabei bildet jedes Element i aus dem Key-Array "k" mit dem $i$ten Eintrag aus dem Value-Array "v" ein Key-Value-Paar. Der erste Tag des Restaurants *Ganesha* ist somit also "addr:city":"Stuttgart". Eindeutig unterscheiden lassen sich die einzelnen Objekte mit Hilfe ihrer einzigartigen ID ("id").

Das Array "p" enthält die Parents des Objekts. Dabei handelt es sich um die IDs der Regionen in denen sich das Objekt befindet. Aufgebaut sind diese Parents laut Bahrdt und Funke (2015) als DAG (directed acyclic graph). Dass das Restaurant *Ganesha* den Parent mit der ID 64 enthält, zeigt somit beispielsweise an, dass es in Deutschland liegt. Parent 570 steht für Baden-Württemberg. Deswegen hat jedes Objekt mit Parent 570 auch den Parent 64, da 64 ein Parent von 570 ist, also Baden-Württemberg in Deutschland liegt. Die Parents helfen also damit, lokale Beziehungen zwischen verschiedenen Objekten festzustellen.

2.1 OSCAR Zellenanordnung

Wie bereits in Kapitel 2 erwähnt, steht die Abkürzung OSCAR für OSm Cell ARrangements (Bahrdt und Funke, 2015). Dieses namensgebende Konzept ermöglicht ein schnelleres Durchsuchen der OSM Daten. Wie in Abbildung 1 dargestellt ist, werden den einzelnen Regionen (in Abbildung 1 links) eine oder mehrere Zellen zugewiesen (Abbildung 1 rechts). Die Zellen werden dann in Form von einem DAG aufgebaut. Zum Beispiel liegt die graue Zelle 11 in Abbildung 1 in der braunen, violetten, orangenen und roten Region, aber nicht in den anderen (Bahrdt und Funke, 2015). Deswegen ist der entsprechende DAG so aufgebaut, dass nur diese Knoten von dem Blattknoten 11 erreicht werden können.

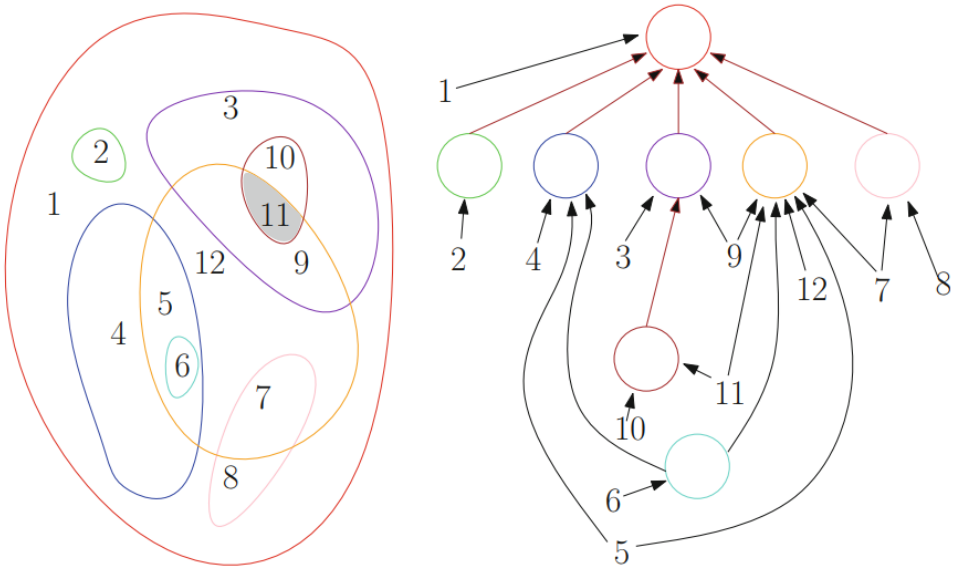


Abbildung 1: Regionen in OSM mit ihrer entsprechenden Zellenrepräsentation (Bahrdt und Funke, 2015).

Jede Zelle wird dabei mit allen Tags versehen, die Objekte in der Region besitzen. Nun können statt den einzelnen Objekten, die Zellen nach den gesuchten Tags durchsucht werden. Das führt zu einer schnelleren Suche, da

die Anzahl an Zellen erfahrungsgemäß deutlich kleiner ist als die Anzahl der Objekte, wie in Abbildung 2 zu sehen ist.

	Items	Cells	Cell depth		Cell sizes	
			max	avg	max	avg
California	5.05 M	9.29 k	7	3.6	335 k	554.6
Germany	39.2 M	152 k	22	8.2	159 k	269.9
Europe	230 M	790 k	24	7.4	846 k	303.6
Planet	357 M	1.23 M	35	6.8	2.32 M	301.1

Abbildung 2: Statistiken zu der OSCAR Zellenrepräsentation (Bahrdt und Funke, 2015).

2.2 OSCAR Query

Für natürlichsprachliche Anfragen an OSCAR muss eine natürlichsprachliche Query in eine OSCAR Query "übersetzt" werden. Dazu wird in diesem Abschnitt der Aufbau einer OSCAR Query vorgestellt. Grundsätzlich wird bei der Eingabe eines einfachen Suchwortes nach Vorkommen dieses Wortes als Substring in Tags eines Objektes gesucht. Außerdem wird geprüft, ob es eine Region im Parents DAG gibt, welche den Substring enthält. Alle Objekte, die sich in einer solchen Region befinden oder den Substring als Teil des Namens beinhalten, werden als Ergebnis auf die OSCAR Query zurück gegeben. Trennt man mehrere Worte einer Query mit einem Leerzeichen, so wird die Schnittmenge der beiden Query Worte gebildet (Bahrdt und Funke, 2015). Mit der Query *restaurant Stuttgart* werden also alle Objekte gesucht, die sowohl den Substring *restaurant*, als auch *Stuttgart* enthalten. Möchte man nach einem Substring mit Leerzeichen suchen, muss das durch einen Backslash vor dem Leerzeichen gekennzeichnet werden.

2.2.1 String Typen

Um anstelle von einer Substringsuche nach dem exakten String zu suchen, muss dieser mit Anführungszeichen versehen sein (Bahrdt und Funke, 2015). Möchte man nach Restaurants in Stuttgart suchen, benutzt man deswegen besser *"Stuttgart" @amenity:restaurant*, da so beispielsweise Ergebnisse im deutlich größeren *Regierungsbezirk Stuttgart* nicht mit berücksichtigt werden. Nach einem Suffix lässt sich suchen, indem dem Suchstring ein Fragezeichen vorangestellt wird. Genauso wird mit einem Fragezeichen hinter dem String eine Präfixsuche erreicht. Damit kann zum Beispiel mit der Query *@amenity:restaurant Stuttgart? ?Vaihingen* nach Restaurants in Stuttgart Vaihingen gesucht werden.

2.2.2 Mengenoperationen

Die Query Sprache von OSCAR wurde mengenbasiert entworfen (Bahrdt, 2013). Das heißt, dass die Ergebnisse einer Query als Menge von gefundenen Objekten betrachtet werden. Sind in der Query zwei Worte von einem Leerzeichen oder von einem von Leerzeichen umschlossenen Slash ('/') getrennt, so entspricht die zurückgegebene Ergebnismenge der Schnittmenge der Mengen, die von den beiden Worten als einzelne Queries gefunden worden wären. Ein von Leerzeichen umschlossenes '−' steht für eine Differenzmenge und ein '+' für eine Vereinigung. Um eine eindeutige Reihenfolge der Ergebnisse festzulegen, können mit Klammern einzelne Teile der Query weiter verschachtelt werden (Bahrdt und Funke, 2015). Symmetrische Differenzmengen können mit dem '^' Symbol erstellt werden.

2.2.3 Objektbezogene Spezifikationen

Mit den bis jetzt vorgestellten Query Spezifikationen werden nach Strings und Substrings gesucht, die sich sowohl in den Regionennamen der Parents als

auch in den Tags der Objekten selber befinden können. Um nur Stringmatches mit den Objekttags zu erhalten, versieht man den entsprechenden Teil der Query mit einem Ausrufezeichen ('!'). Soll nicht nach den Namen der Objekte gesucht werden, sondern nach einer bestimmten Objektart, werden die besten Ergebnisse mit der Eingabe expliziter Tags erreicht. Dabei kann mit *@key* nach bestimmten Keys und mit *@key:value* nach Key Value Paaren gesucht werden. Eine ausführliche Auflistung möglicher Keys und Values findet sich im OpenStreetMap Wiki[5]. Dabei wird auch auf die Anwendungsmöglichkeiten und Bedeutungen der einzelnen Key Value Paare eingegangen.

2.2.4 Lokale Spezifikationen

Genauso wie mit einem Ausrufezeichen direkt die Tags der Objekte durchsucht werden, kann mit einem Hashtag ('#') nach Vorkommen des Strings oder Substrings in den Parents gesucht werden. Dadurch werden gezielt Objekte gesucht, die sich in bestimmten Regionen von OSM befinden. OSCAR bietet jedoch noch einige weitere Möglichkeiten, um die Relationen zwischen einem Objekt und den Regionen oder anderen Objekten genauer anzugeben:

- Mit *in* werden nur Objekte gefunden, die sich in der gesuchten Region befinden.
- Die Queryzusätze *:north-of*, *:south-of*, *:east-of* und *:west-of* geben an, dass nur nach Objekten gesucht werden, die sich entsprechend nördlich, südlich, östlich oder westlich von der gesuchten Region befinden.
- Um einen Bereich zwischen zwei anderen Regionen zu durchsuchen, wird das Schlüsselwort *between* benötigt. Dabei ist zu beachten, dass *between* in dem Query String zwischen den beiden Regionen stehen muss.

[5] `https://wiki.openstreetmap.org/wiki/Map_Features`

- Mit dem Zusatz *near* können Bedingungen angegeben werden, die in der Nähe der Objekte erfüllt sein müssen. Dabei kann es sich um Regionen oder auch um Objekte mit bestimmten Tags handeln.
- Ähnlich dazu kann die Umgebung einer Region oder eines Objektes auch mit dem Prozentzeichen ('%') durchsucht werden. Die genaue Umgebungsgröße lässt sich mit *'%n%'* noch genauer festlegen. Dabei steht das n für den Radius in Kilometern, der bei der Suche berücksichtigt werden soll.

2.2.5 Beispielqueries

Um die komplexe Querysprache von OSCAR besser verstehen zu können, werden in diesem Abschnitt einige Beispielqueries betrachtet. Mit den natürlichsprachlichen Anfragen an OSCAR soll nach ebenso komplexen Zusammenhängen gesucht werden können, damit die Querysprache dadurch nicht eingeschränkt wird.

"Stuttgart" @amenity:restaurant (@cuisine:german + @cuisine:italian) Mit dieser Query kann nach Restaurants mit deutscher oder italienischer Küche in Stuttgart gesucht werden.

@shop:supermarket :east-of #Stuttgart Diese Query kann genutzt werden um Supermärkte zu finden, welche östlich von Stuttgart liegen.

%5%% (@waterway:river "Donau") @tourism:camp_site Nach Campingplätzen, die sich im Umkreis von fünf Kilometern um den Fluss Donau befinden, kann mit dieser Query gesucht werden.

3 Ontologiebasierter Datenzugriff

Um natürlichsprachliche Anfragen an OSCAR zu ermöglichen, wird ein Mapping von natürlichsprachlichen Ausdrücken auf OSM-Tags benötigt. In dieser Arbeit wird dazu eine Ontologie von Codescu et al. (2011a) genutzt. Eine Ontologie ist eine Darstellung von Begrifflichkeiten (Konzepte) und den zwischen diesen Konzepten bestehenden Relationen. Die für diese Arbeit genutzte Ontologie besteht aus zwei aufeinander gemappten Ontologien: einer Ontologie für OSM Tags mit dem Namen OSMonto (Codescu et al., 2011b) und einer Ontologie für räumlich lokalisierte Aktivitäten (Codescu et al., 2011a). Dadurch können den verschiedenen Aktivitäten passende OSM Tags zugeordnet werden.

3.1 OSMonto

OSMonto ist eine Ontologie für OpenStreetMap Tags von Codescu et al. (2011b). Diese Ontologie erleichtert den Überblick über die einzelnen Tags und deren hierarchische Struktur im Vergleich zu der Darstellung der Tags in einem Wiki (Codescu et al., 2012). Dabei soll die Ontologie jedoch möglichst nah an den eigentlichen OSM Tags bleiben.

Um die Key-Value Beziehungen möglichst übersichtlich zu repräsentieren, wird jeder Key als Vaterknoten seiner dazugehörigen Values dargestellt (Codescu et al., 2011b). Ein Problem ist dabei, dass manche Keys und Values den gleichen Namen haben. Die Entitäten der Ontologien müssen jedoch eindeutig sein. Deswegen haben Codescu et al. (2011b) den Keys den Präfix "k_" und Values "v_" hinzugefügt. Ein Ausschnitt aus der OSMonto Ontologie, insbesondere einige Values des Keys *cuisine*, ist in Abbildung 3 zu sehen. Ein weiteres Problem bei der Darstellung der Tags ist, dass einige Values zu mehreren Keys gehören können. Vor allem die Values *yes* und *no* tauchen bei einigen Keys auf. Deswegen wurde zusätzlich ein *yes/no* Knoten eingefügt, welcher eine Unterklasse aller Keys ist, die *yes* und *no* als mögliche Values

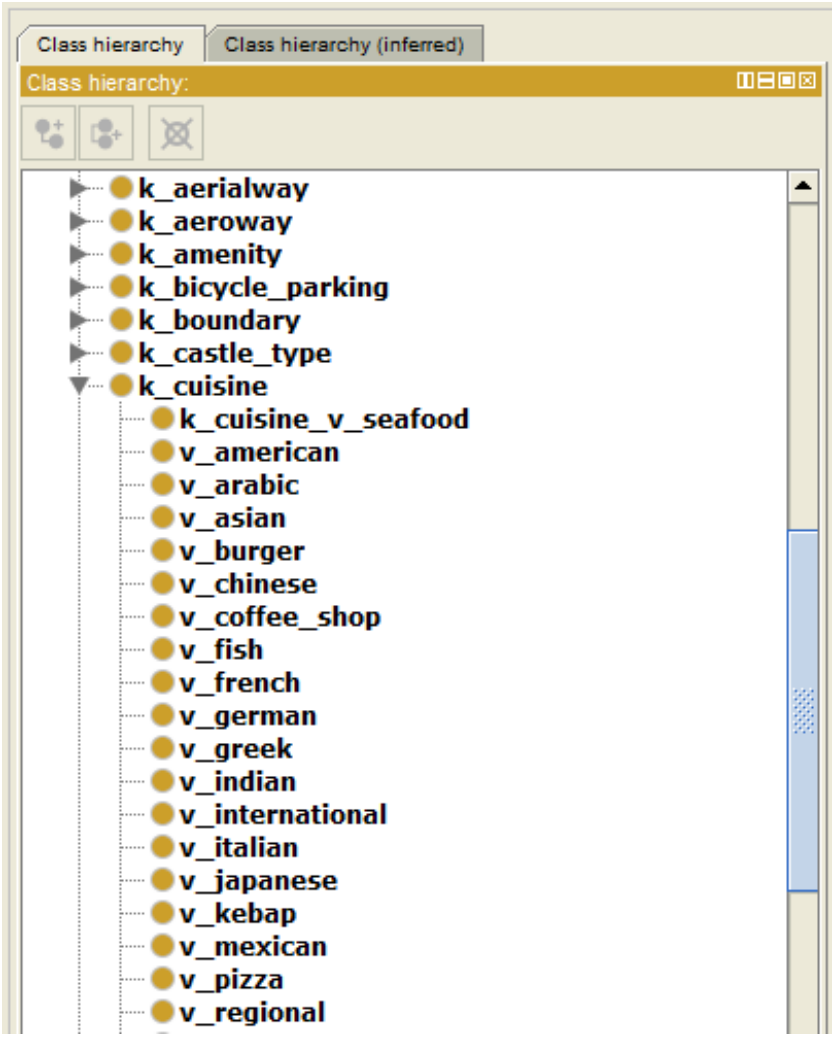


Abbildung 3: Auschnitt der OSMonto Ontologie dargestellt mit Protégé[6].

haben. Für die anderen Values, die öfter vorkommen, wurde der Name des Values um den Key ergänzt. So sieht man zum Beispiel in Abbildung 3 den Value *k_cuisine_v_seafood*. Das liegt daran dass der Name *v_seafood* bereits von einem Value des Keys *k_shop* belegt ist.

Manche Keys haben als Value keine OSM Konstanten (wie beispielsweise die Values *restaurant* oder *university* von dem Key *amenity*) sondern stattdessen einen beliebigen String (zum Beispiel der Name des Objekts) oder eine

[6] `https://protege.stanford.edu`

Zahl (wie beispielsweise die Anzahl der Stellplätze auf einem Parkplatz) (Codescu et al., 2012). In diesem Fall wird dem Key ein Datentyp *string* oder *numeral* zugeteilt, der angibt welchen Typ die Values dieses Keys haben müssen. Zu diesen Keys gehören auch die Keys, welche die Adresse des Objekts angeben. Diese wird nicht in einem, sondern in mehreren Tags angegeben (zum Beispiel durch Keys wie *addr:city*, *addr:street* und *addr:housenumber*). Deswegen haben Codescu et al. (2012) einen Knoten *address* eingefügt, welcher die *addr:* Tags zusammen fasst. Dabei ist jeder Key wieder auf den Datentyp *string* festgelegt, wie in Abbildung 4 zu sehen ist.

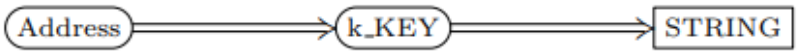


Abbildung 4: Repräsentation von Adressen in OSMonto (Codescu et al., 2012).

Eine große Stärke von OSM ist, dass jeder Objekte taggen kann. Das führt jedoch laut Codescu et al. (2011b) dazu, dass einige Objekte nicht optimal getaggt werden. So kann ein unerfahrener User zum Beispiel einen neuen Tag erstellen, obwohl es schon einen anderen etablierten Tag für ein solches Objekt gibt. Außerdem schleichen sich dadurch immer wieder Rechtschreibfehler in die Tags ein. Daraus resultiert eine große Menge an falschen oder nicht optimalen Tags in den OSM Daten, welche in der Ontologie nicht repräsentiert werden sollen. Um sicherzustellen, dass die Ontologie nur relevante Tags enthält, haben Codescu et al. (2011b) beschlossen lediglich Tags aufzunehmen, die mindestens 100 mal in den OSM Daten vorkommen. Diese Tags wurden durch einige von dem OSM Wiki gelisteten Tags ergänzt. Es handelt sich dabei hauptsächlich um neue Tags, die in der Zukunft genutzt werden oder andere Tags ersetzen sollen. Dadurch soll die Ontologie länger aktuell bleiben (Codescu et al., 2011b).

Die Darstellung der OSM Tags als Ontologie bietet außerdem die Möglichkeit Abhängigkeiten zwischen Keys darzustellen. Ist beispielsweise ein Objekt mit *amenity* = *restaurant* getaggt, so ist es laut Codescu et al. (2012) wahrschein-

lich, aber nicht notwendig, dass auch der Key *cuisine* für dieses Objekt verwendet wird. Dabei unterscheiden Codescu et al. (2012) zwischen zwei Fällen:

- Einige Tags können nur in Gegenwart eines bestimmten anderen Tags genutzt werden. Ein Objekt kann beispielsweise nur mit *theatre:genre = comedy* getaggt werden, wenn es auch mit dem *amenity = theatre* Tag versehen ist. Dies wird, wie in Abbildung 5 dargestellt, durch einen Verweis des Values (in diesem Beispiel *theatre*) auf den Key (hier *theatre:genre*) in OSMonto repräsentiert.

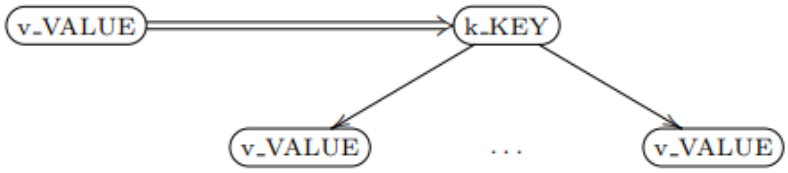


Abbildung 5: Repräsentation von Keys in OSMonto, welche nur mit einem bestimmten Value vorkommen können (Codescu et al., 2012).

- Andere Tags können dagegen in der Gegenwart von mehreren anderen Tags genutzt werden. So können beispielsweise nicht nur Restaurants mit dem *cuisine* Tag versehen werden, sondern auch Cafes oder Biergärten. In diesem Fall wird ein zusätzlicher Knoten eingefügt (in diesem Beispiel *can_have_k_cuisine*) der eine Überklasse der einzelnen Values ist, die diesen Key haben können. Von diesem neuen Knoten *can_have_k_cuisine* wird auf den eigentlichen Knoten *k_cuisine* verwiesen. Dieser Fall wird in Abbildung 6 anschaulich dargestellt.

3.2 Ontologie für räumlich lokalisierte Aktivitäten

Da das System der natürlichsprachlichen Anfragen an OSCAR auch für User ohne OSM Kenntnisse gedacht ist, muss damit gerechnet werden, dass ein solcher User das Tagsystem von OSM nicht kennt. Deswegen muss auch

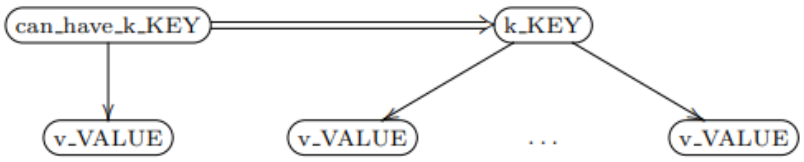


Abbildung 6: Repräsentation von Keys in OSMonto, welche im Zusammenhang mit mehreren anderen Values vorkommen können (Codescu et al., 2012).

nach anderen, natürlichsprachlichen Begriffen gesucht werden können, welche dann auf die OSM Tags gemappt werden. Dazu wird eine zweite Ontologie von Codescu et al. (2011a) verwendet, welche aus Aktivitäten besteht, die an bestimmte Lokalitäten gebunden sind. Diese Ontologie wurde so gestaltet, dass ein automatisches Mapping auf OSMonto möglichst einfach ist (Codescu et al., 2011a).

Das zentrale Konzept der Ontologie ist *Activity* (Codescu et al., 2011a). Die Unterkonzepte davon sind verschiedene Orte, an denen bestimmte Aktivitäten stattfinden können. Besondere Schwerpunkte der Ontologie sind hauptsächlich Orte aus den Bereichen Gesundheitswesen (in der Ontologie durch den Knoten *Health* repräsentiert), Freizeit (*Leisure*), Einkaufen (*Shopping*) und Gastronomie (*Gastronomy*). Aber auch andere Bereiche aus den Kategorien alltägliches Leben und Tourismus sind in der Ontologie abgedeckt (Codescu et al., 2011a).

Abbildung 7 zeigt beispielhaft, wie Restaurants in der Ontologie dargestellt werden. Die Küche eines Restaurant kann entweder durch die Nationalität (zum Beispiel *französische Küche*) oder durch eine bestimmte Essensart (wie zum Beispiel *Pizzeria*) kategorisiert werden (Codescu et al., 2011a). Die verschiedenen Essensarten oder Nationalitäten werden durch Subkonzepte respräseniert, welche in Abbildung 7 als gestrichelte Linien dargestellt werden. Dennoch kann auch einer bestimmten Küchenart eine Nationalität zugewiesen werden. Diese möglichen Rollen werden in Abbildung 7 durch Pfeile zwischen den einzelnen Konzepten dargestellt (Codescu et al., 2011a).

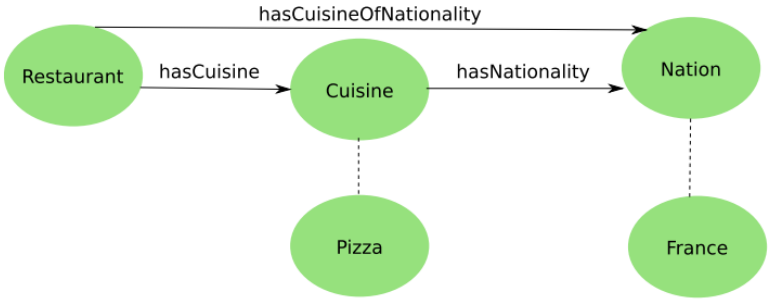
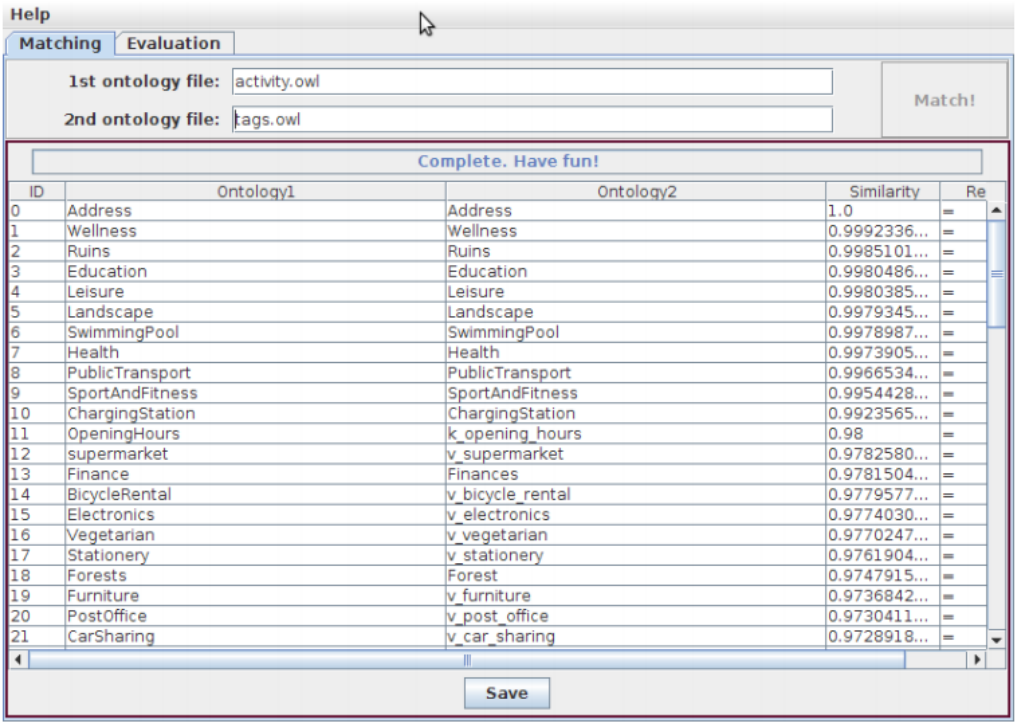


Abbildung 7: Restaurants in der Ontologie für räumlich lokalisierte Aktivitäten von Codescu et al. (2011a).

3.3 Ontologie Mapping

Die beiden in den Kapiteln 3.1 und 3.2 vorgestellten Ontologien wurden von Codescu et al. (2011a) aufeinander gemappt. Dadurch können den verschiedenen Aktivitäten bestimmte OSM Tags zugeordnet werden, welche diese Aktivitäten repräsentieren. Da ein manuelles Mapping relativ umständlich ist, haben Codescu et al. (2011a) das Tool Falcon (Hu und Qu, 2008) genutzt, um Paare von korrespondierenden Konzepten in den beiden Ontologien zu finden.

Ein Teil des Ergebnisses des Matchings mit Falcon ist in Abbildung 8 zu sehen. Falcon bewertet dafür die Ähnlichkeit der Konzepte in den beiden Ontologien zueinander. Dabei wird eine neue Ontologie erstellt in der die ähnlichsten Konzepte zu einem gemeinsamen Konzept zusammengefasst werden. Mit Falcon konnten Codescu et al. (2011a) 80% des Ontologie Mappings automatisch durchführen. Die von Falcon produzierten Paare mussten noch manuell geprüft und teilweise ergänzt werden. Das Resultat ist eine Ontologie mit der Orte für bestimmte Aktivitäten in OSM gesucht werden können (Codescu et al., 2011a).



ID	Ontology1	Ontology2	Similarity	Re
0	Address	Address	1.0	=
1	Wellness	Wellness	0.9992336...	=
2	Ruins	Ruins	0.9985101...	=
3	Education	Education	0.9980486...	=
4	Leisure	Leisure	0.9980385...	=
5	Landscape	Landscape	0.9979345...	=
6	SwimmingPool	SwimmingPool	0.9978987...	=
7	Health	Health	0.9973905...	=
8	PublicTransport	PublicTransport	0.9966534...	=
9	SportAndFitness	SportAndFitness	0.9954428...	=
10	ChargingStation	ChargingStation	0.9923565...	=
11	OpeningHours	k_opening_hours	0.98	=
12	supermarket	v_supermarket	0.9782580...	=
13	Finance	Finances	0.9781504...	=
14	BicycleRental	v_bicycle_rental	0.9779577...	=
15	Electronics	v_electronics	0.9774030...	=
16	Vegetarian	v_vegetarian	0.9770247...	=
17	Stationery	v_stationery	0.9761904...	=
18	Forests	Forest	0.9747915...	=
19	Furniture	v_furniture	0.9736842...	=
20	PostOffice	v_post_office	0.9730411...	=
21	CarSharing	v_car_sharing	0.9728918...	=

Abbildung 8: Mapping der beiden Ontologien mit Falcon (Codescu et al., 2011a).

3.4 Ontologie für lokalisierte Aktivitäten in OSM

Die in Abschnitt 3.3 beschriebene Ontologie von Codescu et al. (2011a), bestehend aus den beiden in Abschnitt 3.1 und 3.2 vorgestellten Ontologien, kann nun genutzt werden, um unterschiedlichen Aktivitäten bestimmte OSM Tags zuzuordnen. Die natürlichsprachlichen Anfragen an OSCAR sollen aus zwei Hauptbestandteilen bestehen:

1. *was*: Spezifiaktion, welche Tags die gesuchten OSM Objekte haben müssen, um von der Query gefunden zu werden.

2. *wo*: Die räumliche Abgrenzung, in der nach Ergebnissen mit den gewünschten Tags gesucht werden soll.

Der *was* Bestandteil wird durch diese Ontologie abgedeckt. Die Konzepte sind dabei die Worte, die in der Query vorgeschlagen und akzeptiert werden. Aus diesen Konzepten werden dann die passenden OSM Tags generiert. Dabei können drei Kategorien von Konzepten unterschieden werden. Je nach Kategorie in der natürlichsprachlichen OSCAR Query werden andere OSM Tags aus den Konzepten abgeleitet:

1. Ist das gesuchte Wort ein OSM Value muss noch der dazugehörige Key ermittelt werden. Dieser ist in der Ontologie immer die Überklasse des Values. Nach diesem Key Value Paar wird dann in OSCAR gesucht. Wird also beispielsweise nach *playground* gesucht, so wird das durch die OSCAR Query *@leisure:playground* dargestellt. Diese Key-Value Beziehung ist in Abbildung 9 anschaulich dargestellt.

2. Wenn es sich bei dem gesuchten Wort um einen OSM Key handelt, kann direkt nach diesem Key in OSCAR gesucht werden. Zum Beispiel wird die Anfrage *leisure* zu *@leisure* übersetzt, da es sich wie in Abbildung 9 gezeigt, bei *leisure* um einen Key handelt.

3. Handelt es sich bei dem gesuchten Konzept um eines, dass ursprünglich von der Ontologie für räumlich lokalisierte Aktivitäten aus Abschnitt 3.2 stammt, sind diese Kozepte Vereinigungen, die aus Keys, Values und anderen Vereinigungskonzepten bestehen können. In Abbildung 10 sieht man beispielsweise das Konzept *Ruins*. Dieses ist eine Vereinigung aus den Values *ruins* des Keys *historic* und *yes* des Keys *ruins*. Das Konzept *Ruins* ist dabei wiederum ein Bestandteil des Konzepts *Attractions*. Wird mit der natürlichsprachlichen Anfrage ein Vereinigungskonzept gesucht, so wird das in der OSCAR Query mit allen Tags aus denen das Konzept besteht dargestellt. Sucht man zum Beispiel nach *Ruins* wird es durch *(@historic:ruins + @ruins:yes)* repräsentiert. Da *Ruins* ein Bestandteil von *Attractions* ist, werden diese Ergebnisse auch bei der Suche nach *Attractions* angezeigt. Zusätzlich zu den Repräsantionen der anderen Bestandteile von *Attractions*.

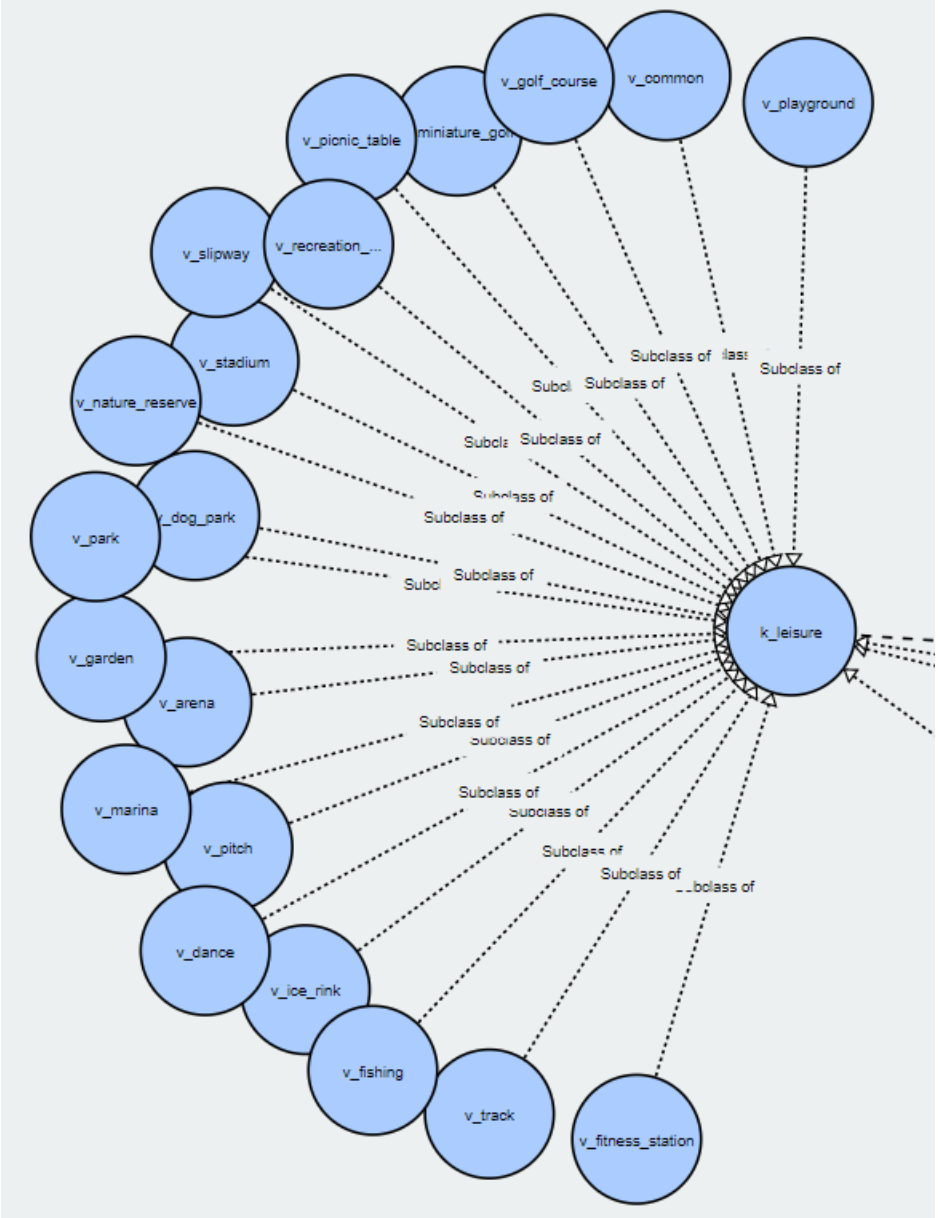


Abbildung 9: Ausschnitt der Darstellung des Keys *leisure* und einigen dazugehörigen Values mit WebVOWL[7].

Dadurch können jedem Konzept der Ontologie für räumlich lokalisierte Aktivitäten OSM Tags zugeordnet werden. Mit diesen Tags sollen die Objekte

[7] `http://vowl.visualdataweb.org/webvowl.html`

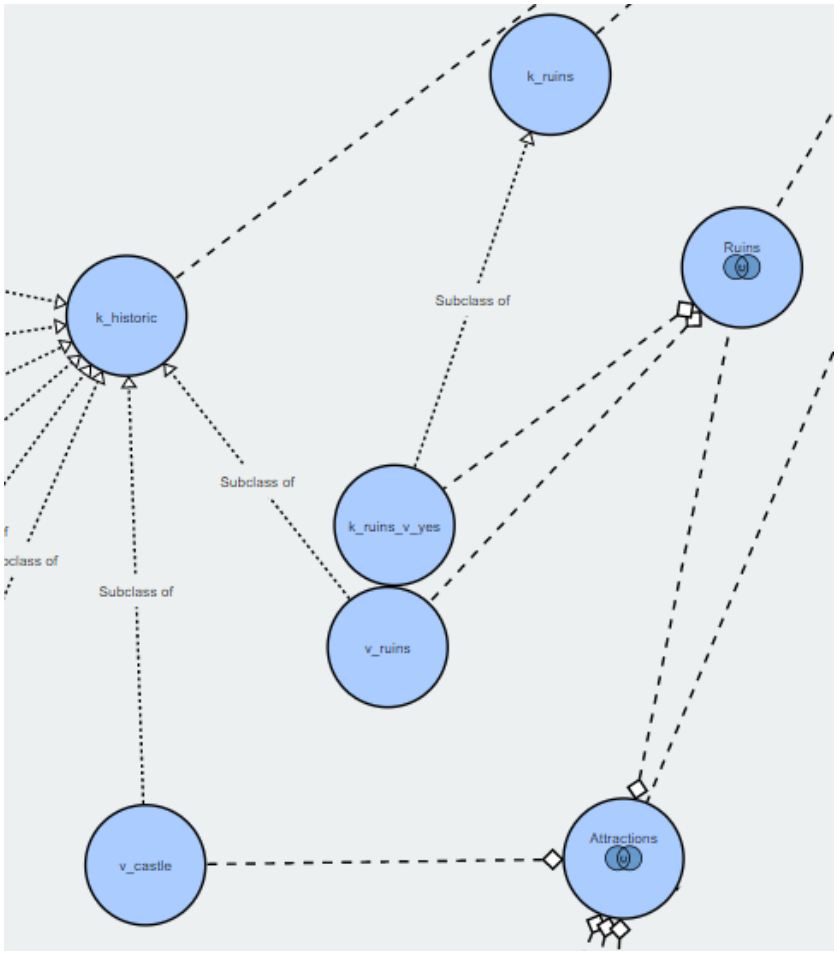


Abbildung 10: Darstellung des Vereinigungskonzepts *Ruins* und dessen Bestandteile mit WebVOWL.

von OSM gefunden werden, die zu dem jeweiligen Konzept passen.

4 Natürlichsprachliche Anfragen

Die Umsetzung von natürlichsprachlichen Anfragen an OSCAR erfolgt durch einen Übersetzer, welcher natürlichsprachliche Anfragen in OSCAR Queries übersetzt. Das Ziel soll dabei in erster Linie eine größere Nutzerfreundlichkeit sein. Ein großes Problem bei der Benutzung von OSCAR ist, dass sich

der Nutzer mit der komplexen Querysprache auskennen muss, um OSCAR effektiv nutzen zu können. Um diesem Problem entgegen zu wirken, benutzt OSCAR Autovervollständigungsvorschläge. Damit muss der Nutzer die Tags nicht genau kennen, sondern kann auf vorgeschlagene Tags zurückgreifen. Auch andere Suchmaschinen, wie zum Beispiel Google[8], benutzen Autovervollständigungen, um eine angenehmere Suche für die Nutzer zu ermöglichen. Da diese Autovervollständigungen so wichtig für die jeweiligen Suchmaschinen sind, werden sie im Folgenden genauer betrachtet.

4.1 Autovervollständigungsvorschläge von OSCAR

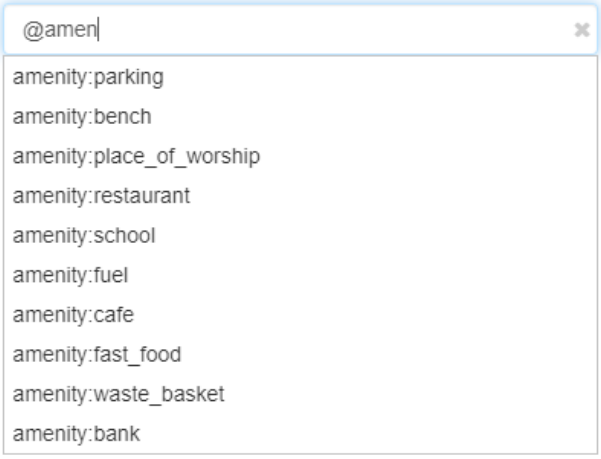


Abbildung 11: Autovervollständigungsvorschläge von OSCAR bei der Eingabe *@amen*[9].

Bei der Eingabe einer Query in OSCAR werden Autovervollständigungsvorschläge angeboten. Diese beschränken sich jedoch auf die Suche von Tags. Es gibt keine Vorschläge für Regionen, Namen oder Verbindungsoperatoren wie Mengenoperationen oder lokale Spezifikationen. Lediglich bei der Eingabe eines Keys, beginnend mit '@' werden Vorschläge gegeben. So ist

[8] `https://www.google.com/`
[9] `https://oscar-web.de/`

beispielsweise in Abbildung 11 zu sehen, dass bei Eingabe von *@amen* Key Value Paare mit dem Key *amenity* vorgeschlagen werden. Auffallend ist dabei, dass die Vorschläge nicht in alphabetischer Reihenfolge gegeben werden. Stattdessen sind diese nach Häufigkeit geordnet. Die vorgeschlagenen Tags stammen von der Website taginfo[10], auf der die häufigsten Tags (rund 2600) aufgelistet sind. Geordnet sind diese Tags danach, bei wie vielen Objekten sie vorkommen. So kommt zum Beispiel der Tag *building:yes*, mit dem über 300 Millionen Objekte getaggt sind, am häufigsten vor. Als zweites folgt *highway:residential* mit über 48 Tausend Objekten. Die passenden Tags von taginfo werden dem Nutzer von OSCAR in dieser absteigenden Reihenfolge vorgeschlagen.

4.2 Google Autocomplete

Die aktuell wohl prominenteste Suchmaschine ist Google. Im Gegensatz zu den Autovervollständigungsvorschlägen von OSCAR, werden von Google nicht nur Vervollständigungen für das aktuelle Wort, sondern für die gesamte Anfrage angeboten, wie in Abbildung 12 zu sehen ist. Das Ziel dieser Autovervollständigungen ist es, schnellere Suchanfragen zu ermöglichen (Sullivan, 2018). Die Vorschläge stammen dabei von einem Algorithmus der mehrere Faktoren berücksichtigt. Unter anderem wird in Betracht gezogen, was und zu welchen Themen der Nutzer zu früheren Zeiten gesucht hat. Außerdem werden aktuelle Suchtrends berücksichtigt. Aus diesem Grund betitelt Sullivan (2018) die Autovervollständigungen explizit als Prognosen ("predicitons") und nicht als Vorschläge ("suggestions"). Der Algorithmus stellt also basierend auf den Daten Vermutungen auf, was der Nutzer am wahrscheinlichsten suchen will.

Dieses System für automatisch generierte Vorschläge hat jedoch auch einige unerwünschte Nebeneffekte. Beispielsweise können Vorschläge mit sexistischem, gewalttätigem oder rassistischem Inhalt angeboten werden. Um das

[10] `https://taginfo.openstreetmap.org/tags`

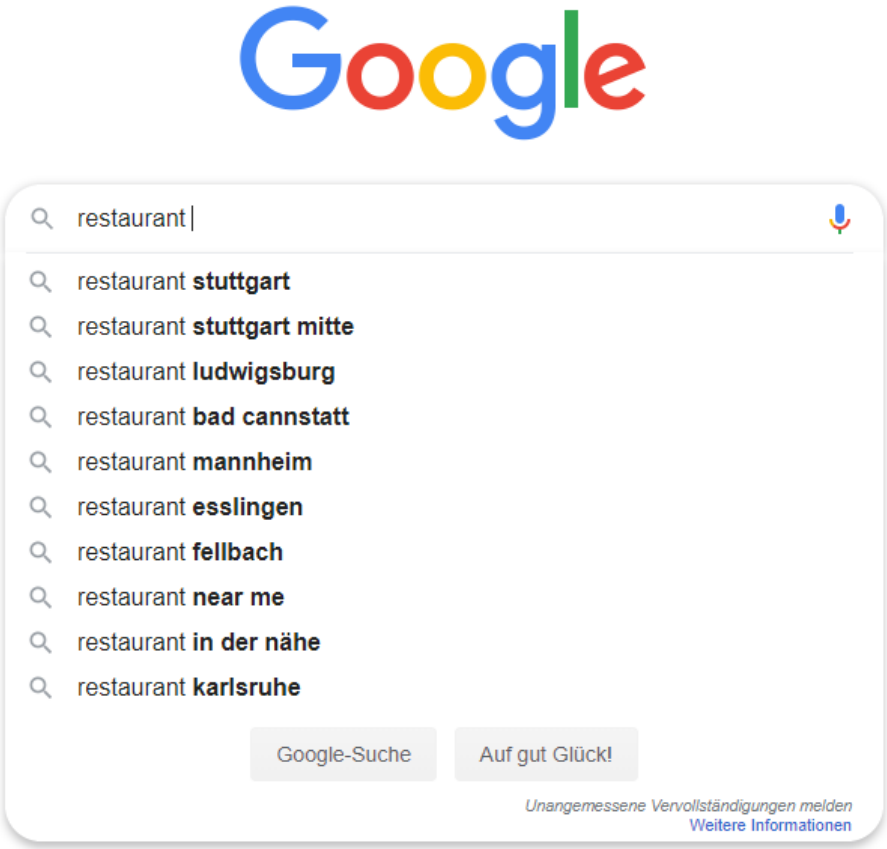


Abbildung 12: Autovervollständigungsvorschläge der Google Suchmaschine bei der Eingabe *restaurant*[11].

zu verhindern, werden solche unangemessene Vorschläge von Google explizit entfernt und verboten (Sullivan, 2018). Dazu macht Google wiederum von den hohen Nutzerzahlen Gebrauch. Mit der laut Sullivan (2018) 2017 eingeführten Reportfunktion für die Nutzer, sollen solche unangemessenen Vorschläge möglichst frühzeitig gefunden und gelöscht werden. Diese Funktion ermöglicht es den Nutzern, wie in Abbildung 13 gezeigt, einzelne Vorschläge mit verschiedenen Begründungen zu melden.

[11] `https://www.google.com/`

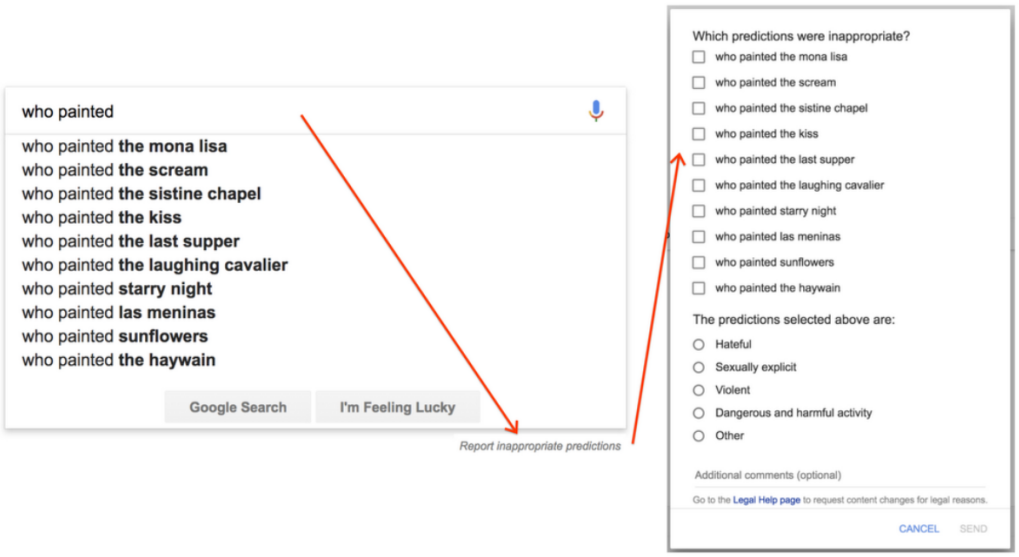


Abbildung 13: Report Funktion zum Melden von unangebrachten Vorschlägen der Google Suchmaschine (Sullivan, 2018).

4.3 Sprachautomat

In den zwei vorangegangenen Abschnitten 4.1 und 4.2 wurden zwei Möglichkeiten für Autovervollständigungsvorschläge präsentiert. Die für den Nutzer hilfreichere Möglichkeit ist die von Google Autocomplete, da nicht nur einzelne Tags, sondern komplette Queries vorgeschlagen werden. Die großen Datenmengen (mehrere Milliarden Suchanfragen pro Tag (Sullivan, 2018)) auf die Google zurückgreifen kann, ermöglichen, dass für fast alle vernünftigen Eingaben sinnvolle Vorschläge angeboten werden. Diese Methode hat jedoch auch zwei beachtliche Nachteile. Zum einen müssen sehr große Datenmengen vorhanden sein. Das heißt insbesondere, dass ein solches System nur mit einer größeren Nutzerzahl gute Ergebnisse produzieren kann. Außerdem muss in ein solches System auch regelmäßig manuell eingegriffen werden, um unerwünschte Vorschläge, wie Vorschläge mit rassistischem oder sexistischem Inhalt, zu verbieten.

Da das System für natürlichsprachliche Anfragen an OSCAR neu entwi-

ckelt wird, sind noch keine Querydaten davon vorhanden. Deswegen wurden für dieses System festgelegte Autovervollständigungswortlisten gewählt. Diese können mit der Zeit aktualisiert werden, falls bestimmte Query Worte hinzugefügt oder entfernt werden müssen. Dennoch können die Vorschläge von Beginn an so festgelegt werden, dass keine unangemessenen Vorschläge angeboten werden. Trotzdem sollen die Autovervollständigungsvorschläge nicht wie bei OSCAR nur Tags vorschlagen, sondern wie bei Google Autocomplete durch die gesamte Query führen. Je nachdem aus welchen Worten die bisherige Query besteht, sollen dementsprechend andere Worte vorgeschlagen werden.

Um diese Idee umzusetzen, wurde der Querysprache der natürlichsprachlichen Anfragen an OSCAR ein deterministischer endlicher Automat (DEA) zugrunde gelegt. Damit gehört sie zur Klasse der regulären Sprachen. Die Idee ist, dass je nachdem in welchem Zustand sich der DEA gerade befindet, Worte aus anderen Listen als Autovervollständigungsvorschläge angeboten werden. Nachdem ein neues Wort eingegeben wurde, legt die Wortliste aus der dieses Wort stammt fest, in welchen Zustand der DEA wechselt.

Für eine bessere Übersicht ist der DEA in Abbildung 14 dargestellt. Wie bereits erwähnt kann die Query in zwei Bestandteile unterteilt werden. Der *was?* Teil wird in den Zuständen eins bis vier gebildet. Die Zustände fünf bis elf bilden den *wo?* Teil der Query. Die Zustände sechs und zehn sind als Endzustände markiert, da in diesen Zuständen die Query abgeschlossen sein sollte. Die Eingabe kann aber auch in jedem anderen Zustand beendet werden. Die ausgehenden Kanten geben die Worte an, die in dem jeweiligen Zustand eingegeben werden können. In Zustand zwei können somit beispielsweise Worte aus den Listen *CONJ, BETWEEN, EXTRA* oder *TRANS* eingegeben werden. Die Liste aus der das eingegebene Wort stammt, legt auch den Zustand fest in den der Automat übergeht. Die einzelnen Listen werden im folgenden genauer betrachtet:

- TAGS: In dieser Liste befinden sich die Elemente der Ontologie aus

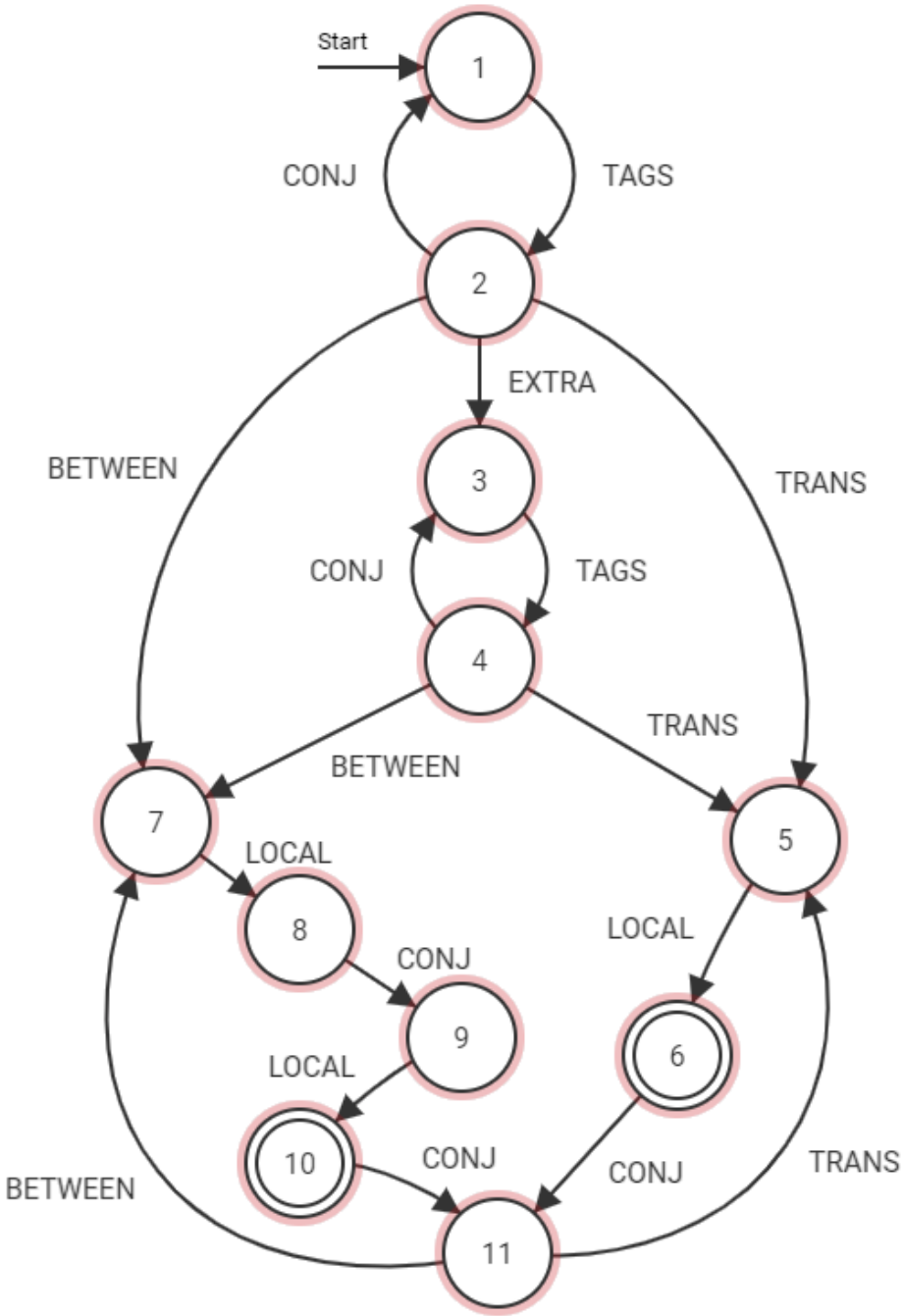


Abbildung 14: Die Querysprache für die natürlichsprachlichen Anfragen an OSCAR als DEA dargestellt.

Abschnitt 3.4. Mit ihnen lassen sich die gewünschten OSM Tags spezifizieren.

- EXTRA: Die Elemente dieser Liste ermöglichen das Hinzufügen einer weiteren Bedingung. Mit *but no* können beispielsweise bestimmte Objekte ausgeschlossen werden. *With nearby* ermöglicht das Suchen nach Objekten mit bestimmten anderen Objekten in der näheren Umgebung.

- CONJ: Diese Liste beinhaltet die Konjunktionen *und* und *oder* und ermöglicht damit die Verknüpfung mehrerer Queries.

- TRANS: Mit TRANS ist die Transition von *was?* zu *wo?* gemeint. Mit Elementen wie *in, near* oder *north of* wird die räumliche Beziehung zwischen den Objekten und den Locations beschrieben.

- BETWEEN: Diese Liste beinhaltet lediglich das Wort *between*. Dabei handelt es sich ebenfalls um eine Transition von *was?* zu *wo?*. Da auf *between* aber immer zwei Locations folgen müssen, wird dieses Schlüsselwort separat behandelt.

- LOCAL: In dieser Liste befinden sich die OSCAR Regionen. Die genauere Zusammensetzung ist im Folgenden in Abschnitt 4.4 beschrieben.

4.4 OSCAR Regionen

Die natürlichsprachlichen Queries an OSCAR bestehen aus zwei Hauptbestandteilen: Nach *was* soll gesucht werden und *wo*, also welche Regionen sollen durchsucht werden. Der *was* Bestandteil wird, wie in Abschnitt 3.4 beschrieben, mit Hilfe von Vorschlägen aus der Ontologie für lokalisierte Aktivitäten in OSM abgehandelt. Um den *wo* Bestandteil abzudecken, soll nach möglichst allen Regionen von OSM gesucht werden können. Da die komplette Liste mit allen Regionen relativ groß ist (ungefähr 1,2 Millionen Regionen), kann der Speicherbedarf durch das Reduzieren dieser Liste deutlich gesenkt

und die Suchgeschwindigkeit gleichzeitig erhöht werden. Durch diese Reduktion der Liste der Regionen soll außerdem die Benutzerfreundlichkeit verbessert werden. Dabei wurden hauptsächlich Elemente mit folgenden Eigenschaften entfernt:

- Doppelte Einträge: Einige Regionen waren mehrfach in der Liste enthalten. Diese Duplikate wurden entfernt.
- Andere Schriftzeichen: Da die natürlichsprachliche Anfragen in Englisch stattfinden, werden Einträge mit Schriftzeichen aus anderen, nicht lateinischen Zeichensystemen, wie zum Beispiel kyrillische, chinesische oder griechische Schriftzeichen, entfernt. Davon ausgenommen sind Zeichen die direkte Varianten der lateinischen Buchstaben sind, wie beispielsweise die deutschen Umlaute, das französische ç oder Akzente.
- Rein numerische Einträge: Einträge, welche lediglich aus Zahlen bestehen wurden ebenfalls entfernt.
- Regionen mit ungewöhnlichem Namen: Manche Regionen haben ungewöhnliche Namen, welche für den Nutzer deswegen schwer zu nutzen sind. Dazu gehören beispielsweise Regionen deren Namen mit einem Leerzeichen beginnen oder mit Klammern oder Anführungszeichen versehen sind. Um die Nutzerfreundlichkeit zu erhöhen, wurden auch solche Regionen aus der Liste entfernt.

Nach diesen Änderungen ist die resultierende Liste deutlich kleiner (ungefähr 737 Tausend statt 1,2 Millionen Regionen) und enthält weniger Einträge, welche für die meisten Benutzer uninteressant und nicht nutzbar sind. Ein Beispiel mit den Autovervollständigungsvorschlägen bei der Eingabe *Stut* sieht man in Abbildung 15. Die ganze, gefilterte Liste aller möglicher Regionen findet man unter `https://fgs.oscar-web.de/Kopfbn/localsFile.js`.

Um die Liste aller Regionen effektiver durchsuchen zu können, wurde sie außerdem alphabetisch sortiert. Dementsprechend liegen auch die weiteren

Abbildung 15: Autovervollständigungsvorschläge von Regionen bei Eingabe *Stut*[12].

Autovervollständigungsvorschläge, wie in Abbildung 15 zu sehen, in alphabetischer Reihenfolge vor.

4.5 Technische Umsetzung

Die Umsetzung des während dieser Thesis erarbeiteten Systems für natürlichsprachliche Anfragen an OSCAR ist auf der Website OscarGui[12] zu sehen. Einen Ausschnitt dieser Seite zeigt Abbildung 16. Darauf sind zwei verschiedene Eingabefelder zu sehen. In das Eingabefeld *Search* kann eine gewöhnliche OSCAR Query eingegeben werden. Die in dieser Thesis vorgestellten natürlichsprachlichen Anfragen können in das Eingabefeld *Natural Language Search* eingegeben werden. Die Eingabe in dieses Feld wird sofort in die OSCAR Query übersetzt, welche automatisch im *Search* Feld angezeigt wird. Wie in Abbildung 17 zu sehen ist, werden bereits eingegebene Teile der natürlichsprachlichen Anfrage direkt in die entsprechenden Repräsentationen in der OSCAR Query Sprache übersetzt. Zu dem Wort, welches noch nicht

[12] `https://fgs.oscar-web.de/`

vollständig geschrieben ist (in diesem Fall *Stutt*), werden Autovervollständigungsvorschläge angeboten.

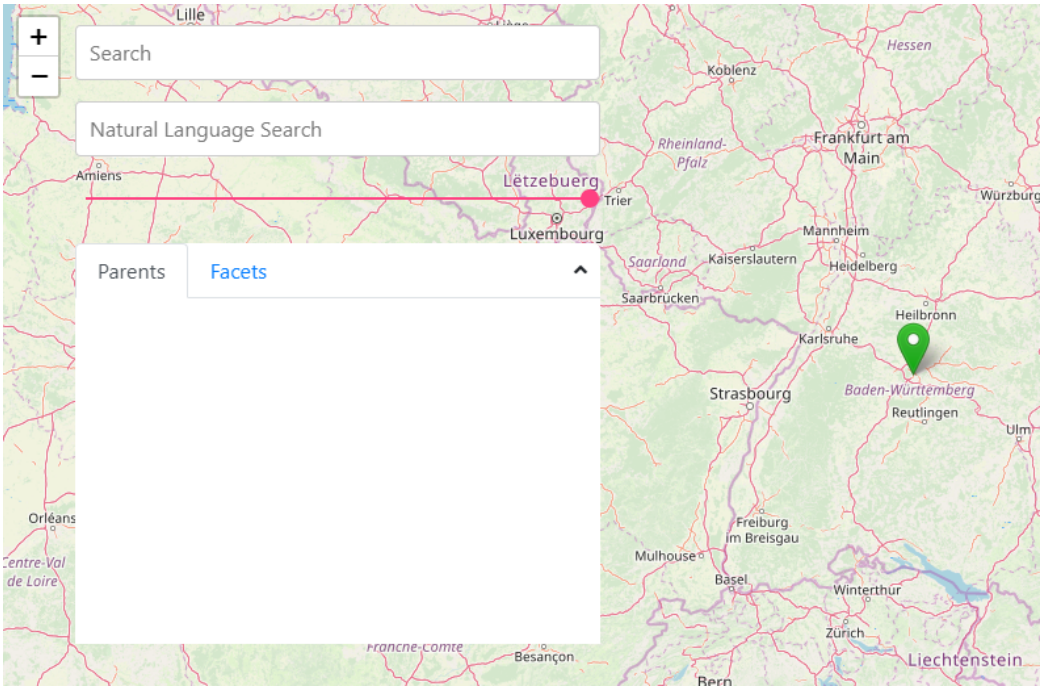


Abbildung 16: Umsetzung der natürlichsprachlichen Anfragen an OSCAR auf der Website OscarGui[13].

Umgesetzt ist das Feature in Form von zwei JavaScript Funktionen. Bei jeder Eingabe eines Zeichens in das *Natural Langauge Search* Textfeld werden diese beiden Funktionen aufgerufen. Die erste Funktion übersetzt den bisherigen Input in eine OSCAR Query und gibt diese zurück. Diese wird dann im *Search* Feld angezeigt. Die zweite Funktion gibt in Abhängigkeit von dem bisherigen Input die Liste mit möglichen Autovervollständigungsvorschlägen zurück. Im genaueren sind diese beiden Funktionen als Pseudocode in Algorithmus 1 und 2 dargestellt. Dabei fällt auf, dass ein Teil des Ablaufs der beiden Funktionen gleich ist. Das liegt daran, dass beides mal der DEA einmal durchlaufen werden muss, um zu wissen welche Zustandsübergänge stattgefunden haben. In Algorithmus 1 muss der Zustand des Automaten bekannt

[13] `https://fgs.oscar-web.de/`

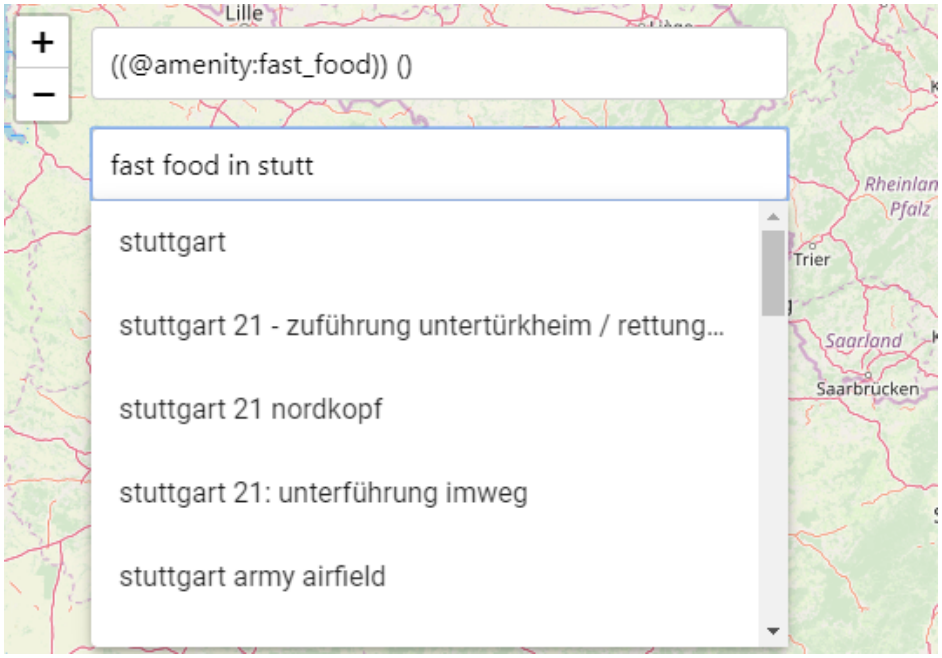


Abbildung 17: Übersetzung einer natürlichsprachlichen Anfrage zu einer OSCAR Query auf der Website OscarGui[14].

sein, um zu überprüfen, ob die Worte aus den von dem DEA vorgegebenen Listen stammen. Deutlich wichtiger ist dies für den Algorithmus 2. Dieser muss am Ende der Eingabe den aktuellen Zustand des Automaten kennen, damit er die richtige Liste (oder Listen) auswählen kann, aus welchen die Autovervollständigungsvorschläge stammen dürfen.

4.6 Problemstellungen

Eine Schwierigkeit bei der Implementierung der beiden Algorithmen aus Abschnitt 4.5 ist das Verhalten bei der Eingabe von Leerzeichen in das *Natural Language Search* Suchfeld. Ein Leerzeichen kann zwei verschiedene Ursachen haben:

- Der aktuelle Ausdruck wurde abgeschlossen. In diesem Fall wechselt

[14] `https://fgs.oscar-web.de/`

Algorithmus 1 Übersetzung von natürlichsprachlichen Anfragen an OSCAR in die OSCAR Query Sprache.

```
String oscarQuery = ""
DEAState = 1
possibleWordsList = words that are accepted by the DEA in state 1
for word in input field do
  \\check for multi word terms first
  if possibleWordsList contains current word + next word(s) then
    translate current word + next word(s) to OSCAR Query language
    oscarQuery.add translation
    change DEAState depending on list the Query term is from
    skip next word(s) in for loop
    \\check for one word terms
  else if possibleWordsList contains current word then
    translate current word to OSCAR Query language
    oscarQuery.add translation
    change DEAState depending on the list the Query word is from
  else
    \\incorrect or incomplete term
    no translation
  end if
  possibleWordsList = words that are accepted by the DEA in current
  state
end for
return oscarQuery
```

Algorithmus 2 Ermittlung der Autovervollständigungsvorschläge.

```
autoCompleteList = empty List
DEAState = 1
possibleWordsList = words that are accepted by the DEA in state 1
for word in input field do
  \\check for multi word terms first
  if possibleWordsList contains current word + next word(s) then
    change DEAState depending on list the Query term is from
    skip next word(s) in for loop
    \\check for one word terms
  else if possibleWordsList contains current word then
    change DEAState depending on the list the Query word is from
  else
    \\check whether it is an incorrect or incomplete term
    if possibleWordsList contains (current word + all subsequent words)
    as prefix of at least one element then
      autoCompleteList.add all words from possibleWordsList with prefix
      (current word + all subsequent words)
      break for loop
    else
      \\incorrect term
      do nothing
    end if
  end if
  possibleWordsList = words that are accepted by the DEA in current
  state
end for
return autoCompleteList
```

der DEA, je nachdem aus welcher Liste der Ausdruck stammt, in einen anderen Zustand. Danach können Worte aus anderen Listen eingegeben werden. Deswegen müssen diese den Autovervollständigungsvorschlägen hinzugefügt werden.

- Der aktuelle Ausdruck besteht aus mehreren Worten und wurde noch nicht abgeschlossen. In diesem Fall findet keine Statusänderung bei dem DEA statt. Die Autovervollständigungsvorschläge müssen deswegen die gleichen bleiben wie zuvor.

Problematisch wird das zum Beispiel, wenn als Location *Stuttgart* eingegeben wird. Da sowohl *Stuttgart* als auch *Stuttgart 21 Nordkopf* in der Liste möglicher Regionen vorkommen. Wird nun nach *Stuttgart* ein Leerzeichen eingegeben muss entschieden werden, von welchen der beiden obigen Fällen ausgegangen werden soll.

Als Kompromisslösung bleibt der DEA in einem solchen Fall erst einmal im aktuellen Zustand, sodass die Locations wie im obigen Beispiel *stuttgart 21 nordkopf* oder *stuttgart army airifield* weiterhin vorgeschlagen werden. Damit der Nutzer jedoch weiß, dass an dieser Stelle auch die Location enden kann und somit auch mit einer Konjunktion aus der CONJ-Liste (siehe Abschnitt 4.3) weiter gemacht werden kann, werden die Konjunktionen *and* und *or* ebenfalls zu den Vorschlägen hinzugefügt. Zur besseren Übersicht werden diese an den ersten beiden Stellen, also noch vor den alphabetisch sortierten Locations, angezeigt. Wird die Eingabe daraufhin mit einer Location fortgeführt, werden die Konjunktionen wieder aus der Liste mit Vorschlägen entfernt. Dementsprechend wird bei Eingabe einer Konjunktion die zuvor aufgeschobene Transition von dem DEA in einen anderen Zustand nachgeholt.

4.7 Nachteile und Einschränkungen

In den vorangegangen Abschnitten von Kapitel 4 wurde die Funktionsweise der natürlichsprachlichen Anfragen an OSCAR genauer erläutert. Nun soll in diesem Abschnitt auf die Nachteile und Schwächen eingegangen werden, die dieses System mit sich bringt. Eine offensichtliche Einschränkung ist, dass nur Worte aus den Listen in das Suchfeld eingegeben werden können. Werden andere Worte eingegeben, so werden diese von dem DEA und somit von der Übersetzung ignoriert. Dies stellt vor allem eine Einschränkung dar, da nicht alle OSM Tags, sondern lediglich die in der Ontologie gelisteten, verwendet werden können. Das System ist außerdem anfällig gegenüber Rechtschreibfehler, da diese nicht korrigiert werden. Stattdessen werden falsch geschriebene Worte, wie andere unbekannte Eingabeworte, von dem DEA ignoriert.

Eine weitere Einschränkung ist, dass der Automat die Reihenfolge der Queryworte vorgibt. Es kann beispielsweise nicht nach *ludwigsburg restaurant*, sondern lediglich nach *restaurant in ludwigsburg* gesucht werden. Zusätzlich erlaubt der DEA keine beliebige Verkettung der Query, wie es natives OSCAR ermöglicht. Es kann zum Beispiel nicht nach *restaurant in ludwigsburg or fast food in stuttgart* gesucht werden, da das von dem Automaten nicht erlaubt wird.

5 Nutzerstudie

Um die Nutzbarkeit der natürlichsprachlichen Anfragen an OSCAR zu evaluieren, wurde eine Nutzerstudie durchgeführt. In dieser wurden die natürlichsprachlichen Anfragen an OSCAR[15] mit nativem OSCAR[16] und Google Maps[17] verglichen. Die Ergebnisse wurden in Form von zwei Standardfragebögen festgehalten, die von den Probanden ausgefüllt wurden: eine Skala für die sub-

[15] `https://fgs.oscar-web.de/`
[16] `http://oscar-web.de/`
[17] `https://www.google.de/maps/`

jektive Usability (SUS) und einen Fragebogen zu der allgemeinen Bewertung der Anforderungen (NASA Task Load Index). Außerdem wurden mit einem Abschlussfragebogen vergleichende Fragen über die drei Systeme gestellt.

5.1 Probanden

An der Nutzerstudie haben 18 Probanden im Alter zwischen 19 und 30 Jahren teilgenommen. Zwei Probanden waren weiblich und 16 männlich. Bis auf zwei belegten außerdem alle einen technischen Studiengang oder haben einen technischen Arbeitsplatz. Alle gaben an, Google Maps mindestens regelmäßig zu nutzen. Zwölf Probanden konnten mit dem Begriff Open Street Map etwas anfangen, sieben gaben sogar an, es selten zu nutzen. Lediglich vier Personen gaben an, von OSCAR schon gehört zu haben. Kein Proband gab an es auch zu nutzen.

5.2 Ablauf

Jeder Proband sollte im Laufe der Studie mit Hilfe von fünf Queries die drei verschiedenen Systeme (OSCAR mit natürlichsprachlichem Ansatz, OSCAR nativ und Google Maps) genauer kennen lernen und anschließend bewerten. Die Reihenfolge in der die Systeme bearbeitet wurden, wurde dabei variiert, sodass alle sechs Kombinationen von jeweils drei Probanden bearbeitet wurden. Als Aufgabe mussten folgende Queries gebildet werden:

- Suchen Sie nach Burger King Restaurants in Stuttgart.
- Suchen Sie nach nicht indischen Restaurants in Ludwigsburg.
- Suchen Sie nach Campingplätzen entlang der Donau.
- Suchen Sie Flughäfen zwischen Stuttgart und München.
- Suchen Sie Kindergärten in Stuttgart Weilimdorf, die eine Schule in der Nähe haben.

Des weiteren wurden die Probanden darauf hingewiesen, dass das Hauptaugenmerk auf das Formulieren der Anfragen gelegt werden soll. Relevant ist besonders, ob mehrere Anläufe für das Formulieren der Studie benötigt wurden und ob das System intuitiv war. Nicht berücksichtigen sollten die Probanden die visuellen Unterschiede der Systeme, die verschiedenen Benutzeroberflächen und die Präsentationen der Ergebnisse. Um den Einstieg in OSCAR zu erleichtern, wurde den Probanden außerdem eine Kurzanleitung zu nativem und natürlichsprachlichem OSCAR bereitgestellt. Auf Wunsch wurden auch Hilfestellungen zu bestimmten Queries gegeben. Genauere Übersicht über diese und weitere für die Studie notwendige Formulare sind in Anhang A.1 zu finden. Bei den Suchanforderungen fällt auf, dass einige so oder in ähnlicher Form auf der Hilfeseite von OSCAR gefunden werden können. Das ist beabsichtigt, da so der Rahmen von 45 Minuten für die Studie besser eingehalten werden kann. Außerdem haben so die Probanden ohne OSM Kenntnisse die Möglichkeit aus den Bausteinen der Hilfeseite einige der Queries ohne weitere Hilfestellung zusammenzustellen, was sonst als schwierig eingeschätzt wurde.

Nachdem ein Proband alle fünf Queries für eines der Systeme formuliert hat, wurden von dem Probanden ein NASA TLX- und ein SUS Fragebogen zu diesem System ausgefüllt. Danach fuhr der Proband mit dem nächsten System fort. Zuletzt wurde ein Abschlussfragebogen (siehe Anhang A.4) ausgefüllt. Mit diesem Fragebogen sollten die Probanden in der Lage sein eine abschließende Bewertung zu geben und die einzelnen Systeme mit Freitexten zu bewerten.

5.2.1 NASA Task Load Index

Der NASA Task Load Index (NASA-TLX) ist eine mehrdimensionale Skala, mit der Anwender die Arbeitsbelastung während oder kurz nach einer Aufgabe bewerten können (Hart, 2006). Die Bewertung findet auf sechs verschiedenen Skalen statt, auf denen jeweils 20 Abstufungen zur Verfügung

stehen:

- geistige Anforderung
- körperliche Anforderung
- zeitliche Anforderung
- Leistung
- Anstrengung
- Frustration

Diese Skalen wurden bei dieser Nutzerstudie gleich stark gewichtet. Es wird somit ein so genanntes *Raw Rating* vorgenommen (Hart, 1986). Die Bewertungen der einzelnen Skalen werden mit fünf multipliziert, sodass das Ergebnis als Prozentwert zwischen 0 und 100 interpretiert werden kann. Der Score eines Systems am Ende der Studie wird aus dem Mittelwert der Scores der sechs Skalen gebildet. Der NASA-TLX Fragebogen ist in Anhang A.2 zu sehen.

5.2.2 System Usability Scale

System Usability Scale (SUS) ist eine Methode für die Bewertung der Usability von Systemen von Brooke (1996). Dabei bewerten Nutzer das System auf den folgenden zehn Skalen mit fünf Abstufungen von starkem Widerspruch bis starker Zustimmung:

- Ich kann mir vorstellen das System regelmäßig zu nutzen.
- Ich empfand das System als unnötig komplex.
- Ich fand das System war einfach zu nutzen.
- Ich denke ich würde die Unterstützung einer technisch versierten Person benötigen, um das System zu nutzen.

- Ich denke die verschiedenen Funktionen des Systems sind gut integriert.
- Ich denke das System enthält zu viele Inkonsistenzen.
- Ich kann mir vorstellen, dass die meisten Leute schnell lernen das System zu nutzen.
- Ich empfand die Bedienung des Systems als sehr umständlich.
- Ich habe mich bei der Nutzung des Systems sehr sicher gefühlt.
- Ich musste viele Dinge lernen bevor ich das System verwenden konnte.

Das Ergebnis des SUS Fragebogens ist ein Score zwischen 0 und 100. Dieser soll einen allgemeinen Überblick über die Usability des jeweiligen Systems geben. Um den Score zu bilden müssen zuerst die Teilwertungen für die zehn Fragen gebildet werden. Bei den Fragen 1,3,5,7 und 9 entspricht die Teilwertung der eingetragenen Position auf der Skala minus eins. Die Teilwertungen für die Fragen 2,4,6,8 und 10 werden mit fünf minus der Position auf der Skala berechnet (Brooke, 1996). Diese zehn Teilwertungen werden addiert und mit 2,5 multiplizert. Dadurch erhält man den endgültigen Score des Systems zwischen 0 und 100. Dabei spricht man bei einem Score von 100 von einem perfektem System ohne Usability Probleme. In Anhang A.3 ist der für diese Studie verwendete SUS Fragebogen zu finden.

5.3 Ergebnisse

In diesem Abschnitt werden die Ergebnisse der Nutzerstudie vorgestellt. Abbildung 18 zeigt wie die Probanden die Anforderungen mit dem NASA-TLX Fragebogen bewertet haben. Dargestellt sind die durchschnittlichen Gesamtscores der drei getesteten Systeme. In Abbildung 19 sind außerdem die durchschnittlichen Bewertungen der einzelnen Skalen dargestellt. Als am wenigsten anspruchsvoll wurde mit einem Score von 26,30% die Suche mit Google Maps bewertet. OSCAR bekam mit 52,41% die höchste Bewertung. Der

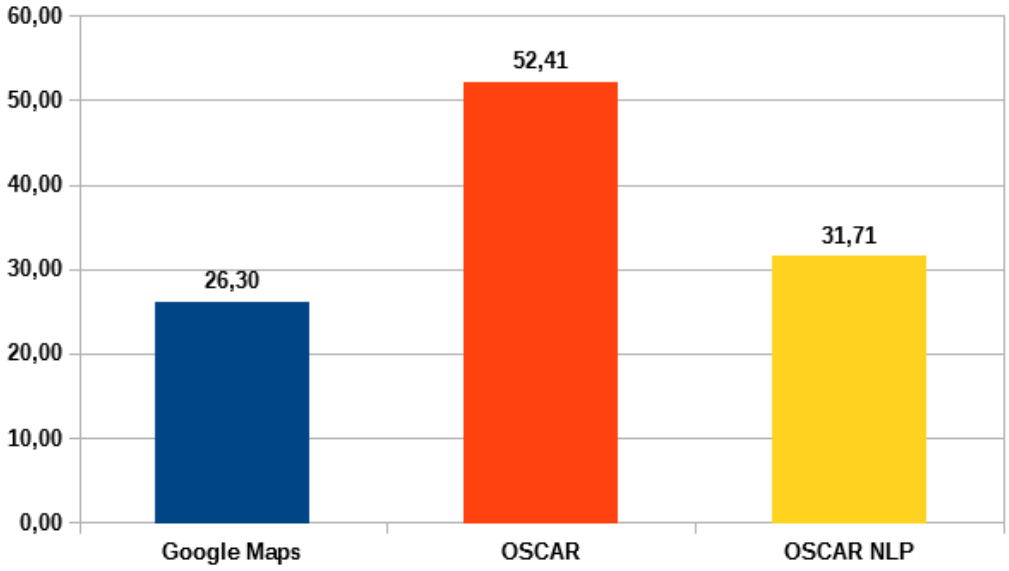


Abbildung 18: Durchschnittliche NASA-TLX Scores von Google Maps, OSCAR und OSCAR mit natürlichsprachlichem Ansatz (NLP).

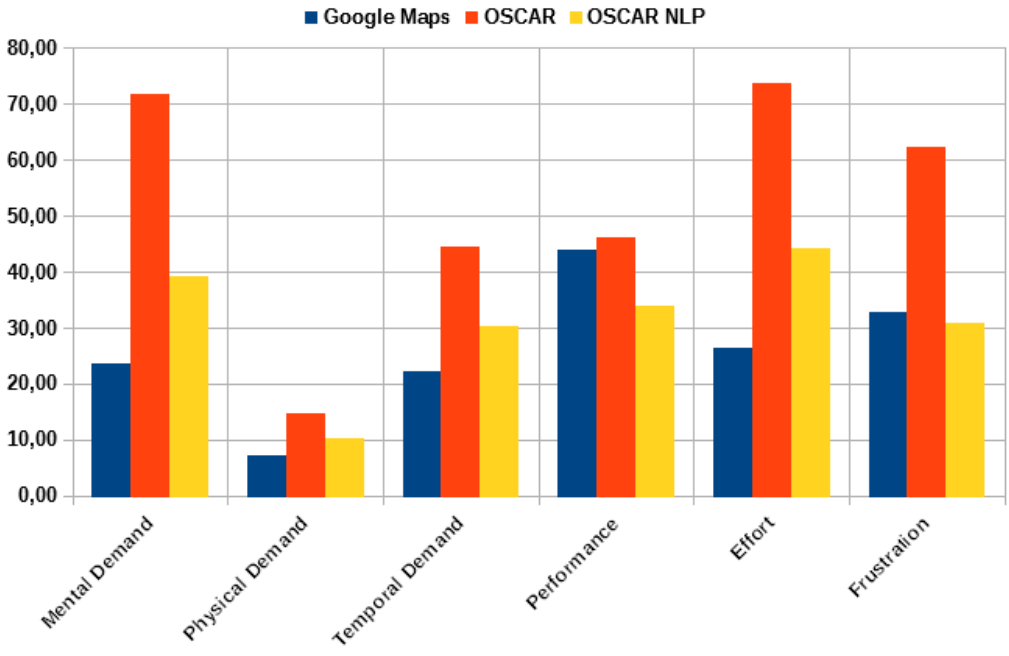


Abbildung 19: Durchschnittliche Scores der einzelnen NASA-TLX Skalen.

natürlichsprachliche Ansatz von OSCAR wurde mit 31,71% bewertet. Die Probanden empfanden das Arbeiten mit diesem Ansatz also als deutlich weniger anspruchsvoll als mit nativem OSCAR. In Abbildung 19 ist erkennbar, dass diese Abstufungen hauptsächlich von den Skalen *Mental Demand, Effort* und *Frustration*, sowie in geringerem Maß von *Temporal Demand* stammen. Die Probanden mussten sich also für OSCAR deutlich mehr anstrengen als für Google Maps. Das liegt zumindest teilweise auch daran, dass alle Probanden schon Erfahrungen mit Google Maps gemacht haben, aber OSCAR zum ersten Mal benutzten. Positiv fällt an Abbildung 19 auf, dass die Probanden mit dem natürlichsprachlichen Ansatz von OSCAR sowohl schneller arbeiten konnten und sich auch gleichzeitig weniger anstrengen mussten. Eine Folge davon ist, dass die Probanden von diesem System deutlich weniger frustriert waren als von OSCAR. Überraschend ist, dass die Frustration im Durchschnitt sogar geringer bewertet wurde als bei Google Maps. Das könnte daran liegen, dass einige der Aufgaben so gestellt waren, dass sie für Google Maps schwer umzusetzen waren. Besonders Queries wie *zwischen Stuttgart und München* und *Kindergärten, die eine Schule in der Nähe haben* sind auf Google Maps sehr schwer oder sogar überhaupt nicht umsetzbar. Daraus folgt auch das zweite überraschende Ergebnis, die Performance. In dieser Kategorie schnitt OSCAR mit natürlichsprachlichem Ansatz deutlich am besten ab.

Die Usability der drei Systeme wurde von den Probanden mit den SUS Fragebögen bewertet. Die Gesamtauswertung ist in Abbildung 20 zu sehen. In Abbildung 21 sind die Auswertungen der einzelnen Skalen im Detail dargestellt. Wie auch schon beim NASA-TLX Fragebogen erreicht Google Maps auch hier den besten Gesamtscore (84,03%). Natives OSCAR hat mit 30,83% die niedrigste Wertung. Wie auch bei der NASA-TLX Wertung liegt OSCAR mit natürlichsprachlichem Ansatz auch bei dem SUS Score zwischen den anderen beiden Ansätzen. Mit 65,51% liegt es aber deutlich näher an Google Maps (18,52 Prozentpunkte) als an nativem OSCAR (34,68 Prozentpunkte). Bei der genaueren Betrachtung der Ergebnisse in Abbildung 21 ist zu

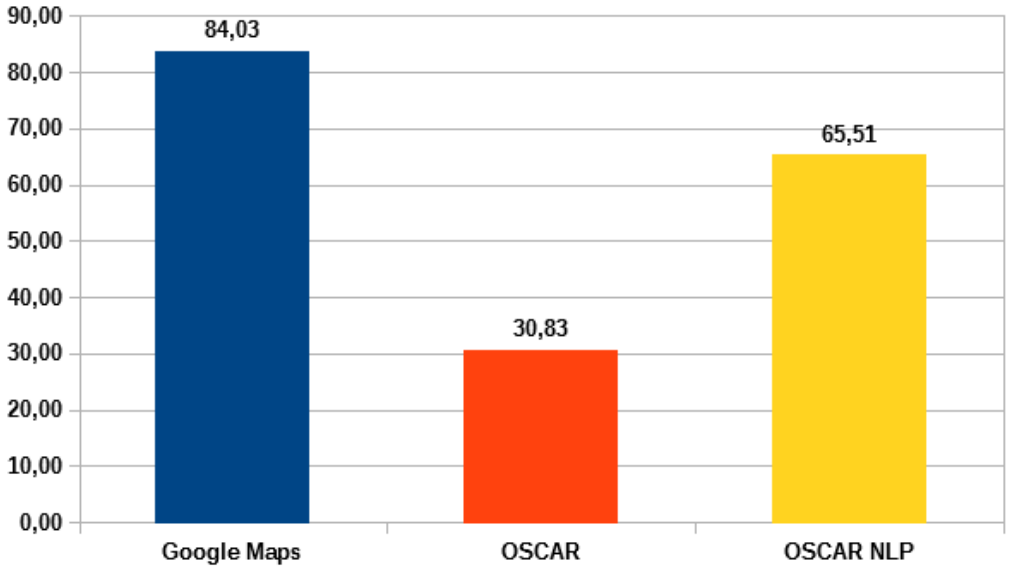


Abbildung 20: Durchschnittliche SUS Scores von Google Maps, OSCAR und OSCAR mit natürlichsprachlichem Ansatz (NLP).

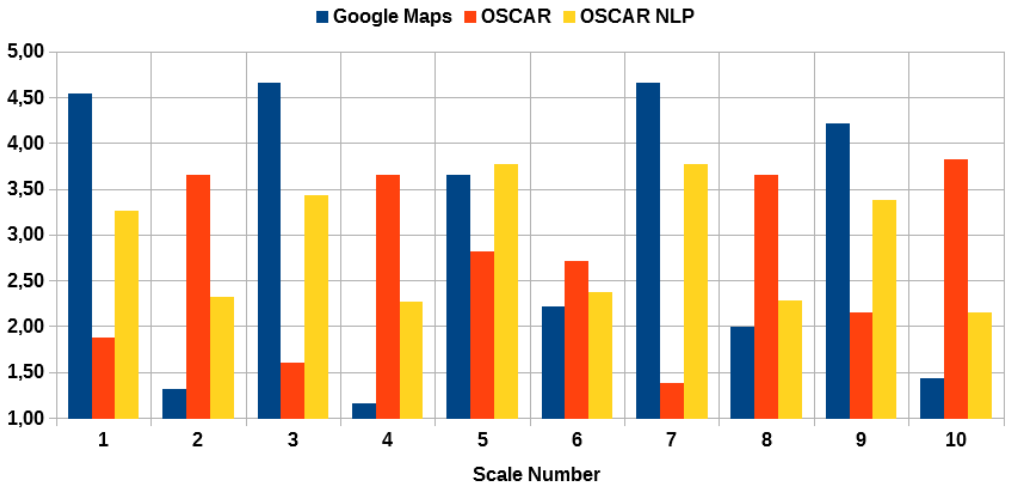


Abbildung 21: Durchschnittliche Antworten der Probanden zu den einzelnen SUS Skalen. Für die Fragen zu den einzelnen Skalen siehe Anhang A.3.

beachten, dass bei den Fragen mit ungerader Nummerierung ein hoher, und bei Fragen mit gerader Nummerierung ein niedriger Wert für gute Usability steht. Dabei fällt auf, dass auf jeder Skala, mit Ausnahme von Skala fünf (*I*

found the various functions in this system were well integrated), der Wert von natürlichsprachlichem OSCAR zwischen den anderen beiden Ansätzen liegt. Im Vergleich zu Google Maps hat OSCAR besonders bei den Fragen drei (*I thought the system was easy to use*), vier (*I think that I would need the support of a technical person to be able to use this system*) und sieben (*I would imagine that most people would learn to use this system very quickly*) schlecht abgeschnitten.

Am Ende der Studie hatten die Probanden mit einem Abschlussfragebogen (siehe Anhang A.4) die Möglichkeit die einzelne Systeme mit Freitextantworten zu bewerten. Außerdem wurde gefragt für welche Suchanfragen sie OSCAR mit natürlichsprachlichem Ansatz besonders geeignet hielten. Dabei wurden am häufigsten Queries mit mehreren verknüpften Anforderungen, wie die räumliche *zwischen Stuttgart und München* Bedingung in Aufgabe vier und *Kindergärten, die eine Schule in der Nähe haben* in Aufgabe fünf genannt. Etwas weniger wurde noch die Möglichkeit genannt, bestimmte Dinge auszuschließen, wie zum Beispiel in Aufgabe 2 nach *nicht indischen Restaurants* gesucht werden sollte. In der zweiten Frage gaben alle bis auf vier Probanden an, bei Google Maps keine, bei nativem OSCAR viel und bei natürlichsprachlichem OSCAR wenig Hilfe benötigt zu haben. Von den anderen vier gaben zwei Probanden an, auch bei Google Maps wenig Hilfe benötigt zu haben, die anderen Beiden gaben bei nativem OSCAR wenig Hilfe an.

Im Folgenden werden einige interessante und öfters genannte Antworten auf die Freitext-Fragen präsentiert. Bei Google Maps wurde vor allem die intuitive und einfache Benutzung hervorgehoben. Besonders einfachere Suchanfragen können leicht mit lediglich Stichworten umgesetzt werden. Außerdem werden zu jeder Zeit hilfreiche Autovervollständigungsvorschläge gegeben. Als negativ nannten die Probanden, dass einige komplexere Anfragen schwierig oder gar nicht umzusetzen waren. Beispielsweise die Suche auf bestimmte Regionen oder auf Objekte in der Umgebung anderer Objekte zu beschränken, ist schwierig. Auch angemerkt wurde, dass Google Maps zum

Beispiel bei *Campingplätz entlang der Donau* lediglich solche fand, die *Donau* im Namen haben.

Als Vorteile von nativem OSCAR wurde genannt, dass sehr genaue und detaillierte Anfragen möglich sind. Da man die Query beliebig verschachteln kann, gibt es viele Möglichkeiten die Anfragen zu formulieren. Somit lassen sich mit der richtigen Query genauere Ergebnisse erzielen als mit Google Maps. Der am öftesten genannte Nachteil ist die komplizierte, ungewohnte Syntax, in die sich neue Nutzer erst einarbeiten müssen. Außerdem müssen die Tags bekannt sein oder erst gesucht werden, um OSCAR nutzen zu können. Erschwert wird das dadurch, dass einige Tags nicht intuitiv sind. So ist zum Beispiel der Tag für Flughäfen *aeroway:aerodrome* und nicht, wie viele Probanden erwartet haben, *amenity:airport*.

Bei den Vorteilen für den natürlichsprachlichen Ansatz von OSCAR nannten die Probanden vor allem, dass die Queryformulierung leichter zu erlernen sei als die von nativem OSCAR. Auch komplexere Anfragen können relativ einfach formuliert werden. Trotz der leichteren Query erreicht man jedoch die gleichen ausführlichen Ergebnisse von OSCAR. Einige erwähnten auch die Möglichkeit der Freitext-Eingabe mit Anführungszeichen positiv. Ein Proband stellte fest, dass man durch die Vorschläge und die begrenzte Liste mehr ausprobieren und so eventuell die richtige Query finden kann, wohingegen man bei nativem OSCAR auf Hilfe angewiesen ist. Als Schwäche nannten die Probanden unter anderem, dass es keinerlei Tooltipps oder Hilfestellung von der Plattform gibt. Außerdem gibt es bei den Vorschlägen ähnliche Worte, die zu ganz anderen Resultaten führen. Zum Beispiel stammen *with nearby* und *close to* aus verschiedenen Listen, weswegen der Automat je nach Wort in einen anderen Zustand übergeht. Durch den Automat muss die Query auch in einer bestimmten, vorgeschriebenen Reihenfolge stattfinden. Diese beiden Punkte machen das System anfällig, sodass bei kleinen Fehlern keine oder falsche Ergebnisse gefunden werden. Als weiterer Punkt wurde angemerkt, dass man in den Möglichkeiten was man suchen kann dadurch eingeschränkt wird, dass nur Worte aus den Vorschlagslisten eingegeben werden können.

Insgesamt zeigt die Studie, dass einige Ziele der natürlichsprachlichen Anfragen an OSCAR umgesetzt werden konnten. Neue Nutzer benötigen weniger Einarbeitungszeit als bei nativem OSCAR, um komplexe Queries bilden zu können. Sie sind in der Lage das System schneller zu nutzen und sind dadurch weniger frustriert. Trotzdem hat das System Inkonsistenzen, welche bei neuen Nutzern für Verwirrung sorgen können. Natürlich ist die Bedienung, vor allem wegen den Restriktionen des Sprachautomats, schwieriger als beispielsweise die von Google Maps, dennoch haben die Probanden festgestellt, dass komplexere, verknüpfte Queries formulierbar sind, die mit Google Maps auch nicht ohne weiteres umzusetzen sind.

6 Zusammenfassung und Ausblick

Ziel dieser Arbeit war das Entwickeln eines Übersetzers, der eine möglichst natürlichsprachliche Anfrage in die OSCAR Syntax übersetzt. Dadurch sollen vor allem neue Nutzer, auch ohne Erfahrung mit OSCAR oder OSM, in der Lage sein die Plattform zu nutzen. In diesem Kapitel werden die Ergebnisse der Arbeit zusammengefasst und evaluiert, in wie weit diese den Erwartungen entsprechen. Zuletzt werden noch mögliche Anknüpfungspunkte für zukünftige Arbeiten vorgestellt.

6.1 Zusammenfassung

Am Anfang der Arbeit wurde die Kartenplattform OSCAR vorgestellt. Dabei wurde besonders die Querysprache genauer analysiert. Danach wurde eine Ontologie vorgestellt, die es erlaubt räumlich lokalisierte Aktivitäten direkt auf OSM Tags zu mappen. Die in dieser Ontologie gelisteten Aktivitäten werden verwendet um den *was?* Teil der natürlichsprachlichen Anfragen an OSCAR abzudecken. Als mögliche Vorbilder für Autovervollständigungsvorschläge wurden als nächstes die Vorschläge von OSCAR sowie Google Autocomplete näher betrachtet und analysiert. Das Ergebnis davon ist ein DEA,

der die Anfragen lenkt und je nach aktuellem Zustand andere Worte als Autovervollständigung vorschlägt. Anschließend wurde erklärt wie der *wo?* Teil der OSCAR Query mit Hilfe von OSCAR Regionen umgesetzt wurde. Darauf folgte eine Übersicht darüber wie die natürlichsprachlichen Anfragen an OSCAR technisch umgesetzt wurden, gefolgt von einer kritischen Evaluation des Systems. Zuletzt wurde die durchgeführte Nutzerstudie vorgestellt und die Ergebnisse dieser Studie analysiert.

Die Ergebnisse der Nutzerstudie deuten darauf hin, dass die natürlichsprachlichen Anfragen an OSCAR das Ziel erreicht haben, komplexe OSCAR Queries intuitiver umzusetzen. Die Probanden konnten mit diesem Ansatz schneller und effizienter arbeiten als mit nativem OSCAR. Dennoch gibt es noch einige Inkonsistenzen und Einschränkungen, welche dem System noch Potenzial zur weiteren Verbesserung bieten.

6.2 Ausblick

Der Sprachautomat ist mit seinen aktuell zwölf Zuständen relativ begrenzt. Er könnte zum Beispiel durch Zustände erweitert werden, in denen sowohl Tags als auch Locations eingeben werden können. Damit könnte das Problem gelöst werden, dass ähnliche Ausdrücke wie *with nearby* und *close to* zu verschiedenen Zuständen führen. Das würde aber als neues Problem hervorrufen, dass es Worte wie beispielsweise *school* gibt, die sowohl in den Tags als auch in den Locations vorkommen.

Eine weitere interessante Untersuchung wäre, die Autovervollständigungsvorschläge anders zu sortieren. Bei den aktuellen, alphabetisch sortierten, Vorschlägen von Locations erscheint das gewünschte Wort erst nachdem einige Buchstaben getippt wurden. Eine Sortierung, welche beispielsweise nähergelegene oder größere Locations bevorzugt, könnte für eine bessere User Experience sorgen.

Besonders für unerfahrene Nutzer wäre eine Dokumentation oder Hilfe-

funktion wie auf der OSCAR Webseite[18] nützlich, damit diese die Möglichkeit haben, sich eigenständig mit dem System vertraut zu machen.

[18] http://oscar-web.de/

A Anhang

A.1 Allgemeine Dokumente der Studie

Studie über natürlichsprachliche
Anfragen an OSCAR

Aufbau:

Die Studie ist auf ca. 45 Minuten angesetzt und folgendermaßen aufgebaut:

1. Aufklärung zum Ablauf und den Bedingungen der Studie
2. Allgemeine Fragen an den Probanden
3. Durchführung von Suchanfragen an verschiedene Kartendienste mit 3 Methoden (OSCAR, OSCAR mit natürlichsprachlichen Anfragen und Google Maps)
4. Ausfüllen des Fragebogens zur Usability der jeweiligen Methode
5. Allgemeine, methodenübergreifende Fragen

Rechte der Probanden:

Die Teilnahme an der Studie ist freiwillig. Sie haben jederzeit das Recht die Studie ohne Folgen zu pausieren oder abzubrechen.

Sie sind außerdem berechtigt Antworten auf einzelne Fragen zu verweigern. Die während dieser Studie gesammelten Daten werden für wissenschaftliche Zwecke gesammelt und vertraulich und anonym behandelt.

Vergütung:

Für die Teilnahme an der Studie erhalten Sie nach der Studie eine Aufwandsentschädigung in Höhe von 10€. Der Erhalt darüber muss mit Name und Unterschrift quittiert werden. Diese Daten werden ausschließlich für die interne Abrechnung verwendet. Es werden keinerlei Verknüpfungen zu den in der Studie aufgenommenen Daten hergestellt.

Mit meiner Unterschrift bestätige ich, dass ich die Informationen gelesen habe und den Bedingungen dieses Formulars zustimme.

_____ _____

Ort, Datum Unterschrift

Allgemeiner Fragebogen

Alter: _____

Geschlecht: _____

Studiengang/Beruf: _____

Ist Ihnen Google Maps ein Begriff? ☐ Ja ☐ Nein
Falls Ja: Wie oft nutzen Sie Google Maps?
☐ (fast) täglich ☐ regelmäßig ☐ selten ☐ nie

Ist Ihnen OpenStreetMap ein Begriff? ☐ Ja ☐ Nein
Falls Ja: Wie oft nutzen Sie OpenStreetMap?
☐ (fast) täglich ☐ regelmäßig ☐ selten ☐ nie

Ist Ihnen OSCAR ein Begriff? ☐ Ja ☐ Nein
Falls Ja: Wie oft nutzen Sie OSCAR?
☐ (fast) täglich ☐ regelmäßig ☐ selten ☐ nie

Aufgaben

In dieser Studie sollen die Suchanfragen an 3 verschiedene Kartendienste miteinander verglichen werden. Ihre Aufgabe ist es, mit den 3 unterschiedlichen Methoden nacheinander jeweils 5 Suchanfragen durchzuführen. Anschließend sollen für jede Methode Fragebögen darüber ausgefüllt werden um diese Methode zu bewerten.

Dabei soll besonders auf folgende Punkte geachtet werden:

- Wie schwer war das Erstellen/Formulieren der Anfrage?
  (War es intuitiv, haben Sie Hilfe benötigt, wie viele Anläufe haben Sie gebraucht?)
- Entsprechen die Suchergebnisse den Erwartungen?
  (Sind falsche/ungewollte Ergebnisse bei den Ergebnissen? Sind es zu wenige Ergebnisse, fehlt eventuell ein Teil?)

Folgende Punkte sollen für diese Studie nicht berücksichtigt werden:

- Die visuellen Unterschiede der Systeme und der Benutzeroberflächen.
- Die Art, wie die Ergebnisse präsentiert werden.
  (Übersichtlichkeit und Strukturierung der Ergebnisse)

Die Aufgaben lauten:

1. Suchen Sie nach Burger King Restaurants in Stuttgart.
2. Suchen Sie nach nicht indischen Restaurants in Ludwigsburg.
3. Suchen Sie nach Campingplätzen entlang der Donau.
4. Suchen Sie Flughäfen zwischen Stuttgart und München.
5. Suchen Sie Kindergärten in Stuttgart Weilimdorf, die eine Schule in der Nähe haben.

Erläuterungen und Hilfestellung

OSCAR natürlichsprachlich:

Von diesem System werden Vorschläge gegeben aus denen die Suchanfragen zusammengestellt werden sollen. Nur Worte aus den Vorschlagslisten werden dabei akzeptiert. Die Querysprache ist Englisch.

An manchen Stellen werden auch freie Eingaben angeboten. Diese Stellen sind mit “free Text“ markiert. An diesen Stellen können umschlossen von Anführungszeichen beliebige Worte stehen. Das ist besonders Geeignet, wenn nach bestimmten Namen gesucht wird.

OSCAR:

Gesucht werden kann mit Key Value Paaren von OpenStreepMap. Nach diesen kann in der Taginfo Datenbank gesucht werden. Die beliebtesten Tags werden auch von OSCAR vorgeschlagen.

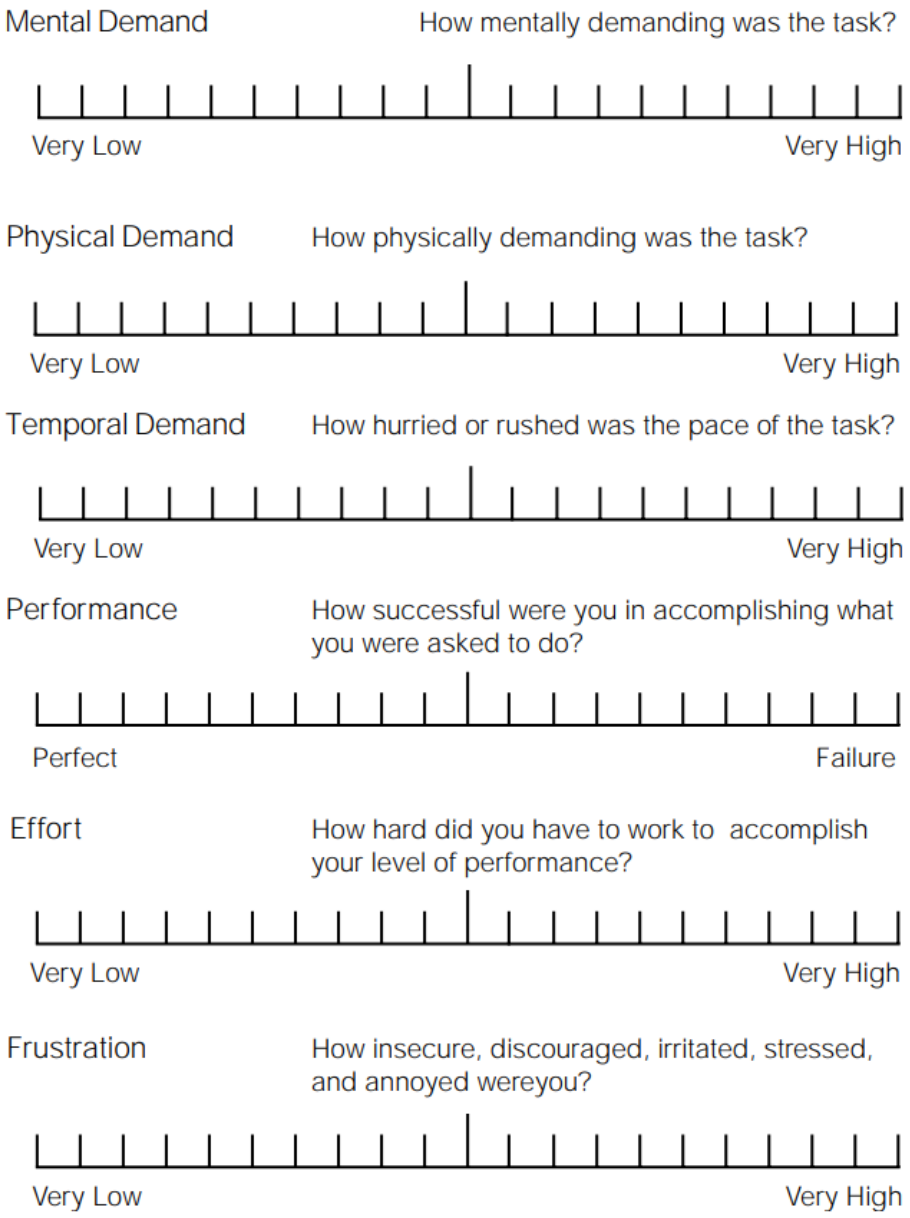
In OSCAR müssen diese Tags folgendermaßen eingegeben werden: @key:value

Bei der Eingabe von normalen Worten wird nach Objekten mit diesem Namen oder Objekten in Locations mit diesem Namen gesucht.

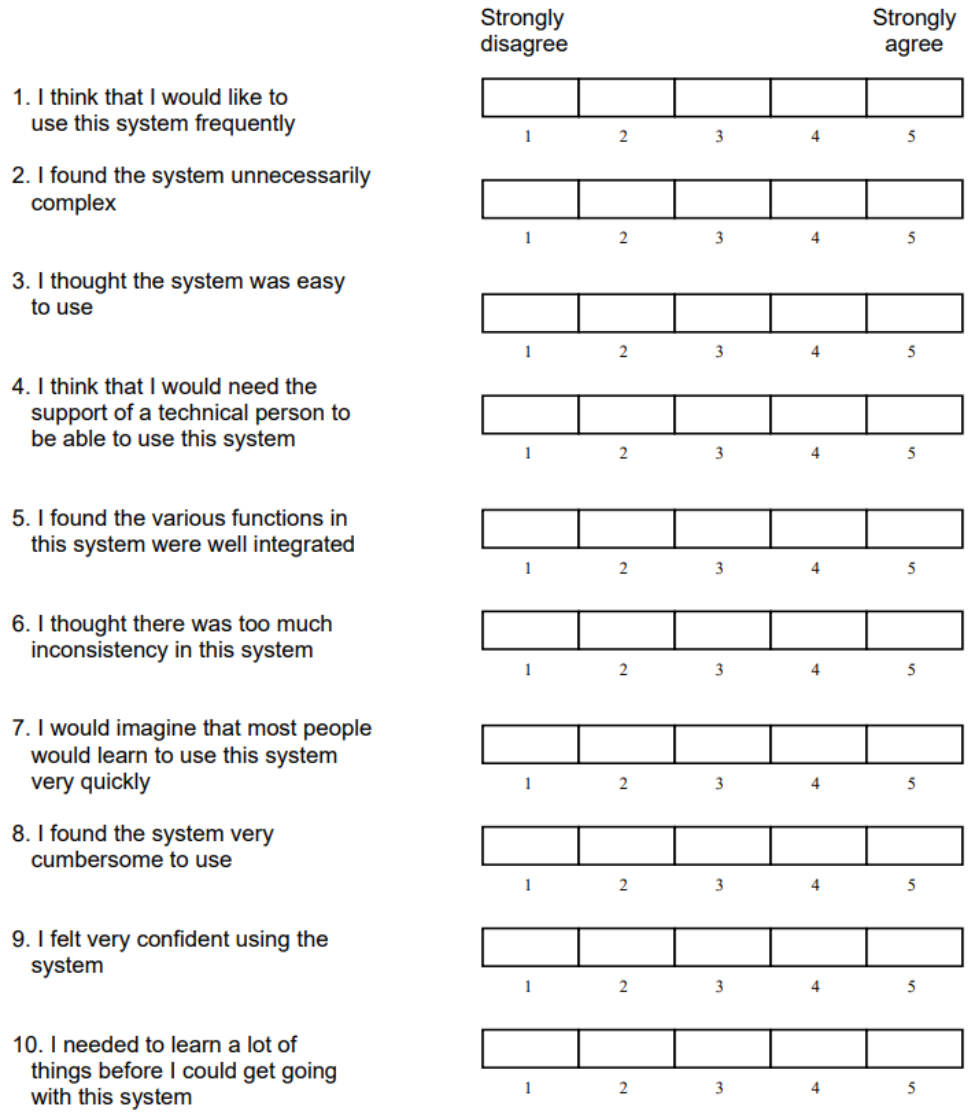
Normalerweise handelt es sich um eine Substring Suche. Um nach exakten Strings zu suchen muss man diese mit Anführungszeichen umschließen.

Unter dem Fragezeichen bietet OSCAR weitere Hilfestellungen sowie beispielhafte Suchanfragen.

A.2 NASA TLX Fragebogen



A.3 SUS Fragebogen



A.4 Abschlussfragebogen

Abschlussfragebogen

Für welche Suchanfragen eigneten sich die natürlichsprachlichen Anfragen an OSCAR besonders? _____

Wie viel Hilfe benötigten Sie für die verschiedenen Systeme? (Inklusive dem beigelegten Hilfe Blatt)

- Google Maps ☐ keine ☐ wenig ☐ viel
- OSCAR ☐ keine ☐ wenig ☐ viel
- OSCAR natürlichsprachlich ☐ keine ☐ wenig ☐ viel

Welche Stärken und Schwächen von Google Maps sind Ihnen aufgefallen?

Welche Stärken und Schwächen von OSCAR sind Ihnen aufgefallen?

Welche Stärken und Schwächen von OSCAR mit natürlichsprachlichen Anfragen sind Ihnen aufgefallen?

Literatur

Daniel Bahrdt. osmfind: Fast Textual Search on OSM Data – On Smartphones and Servers. In *Proceedings of the Second ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, pages 35–42. ACM, 2013.

Daniel Bahrdt und Stefan Funke. OSCAR: OpenStreetMap Planet at Your Fingertips via OSm Cell ARrangements. In *International Conference on Web Information Systems Engineering*, pages 153–168. Springer, 2015.

John Brooke. SUS - A Quick and Dirty Usability Scale. *Usability Evaluation in Industry*, 189(194):4–7, 1996.

Mihai Codescu, Gregor Horsinka, Oliver Kutz, Till Mossakowski, und Rafaela Rau. DO-ROAM: Activity-Oriented Search and Navigation with OpenStreetMap. In *International Conference on GeoSpatial Sematics*, pages 88–107. Springer, 2011a.

Mihai Codescu, Gregor Horsinka, Oliver Kutz, Till Mossakowski, und Rafaela Rau. Osmonto - an Ontology of OpenStreetMap Tags. *State of the Map Europe (SOTM-EU)*, 2011, 2011b.

Mihai Codescu, Daniel Couto Vale, Oliver Kutz, und Till Mossakowski. Ontology-based Route Planning for OpenStreetMap. In *Terra Cognita@ ISWC*, pages 62–73, 2012.

Sandra G Hart. NASA Task Load Index (TLX) 1.0: Paper and Pencil Package. 1986.

Sandra G Hart. NASA-Task Load Index (NASA-TLX); 20 Years Later. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 50, pages 904–908. Sage Publications Sage CA: Los Angeles, CA, 2006.

Wei Hu und Yuzhong Qu. Falcon-AO: A Practical Ontology Matching System. *Journal of Web Semantics*, 6(3):237–239, 2008.

Danny Sullivan. How Google Autocomplete Works in Search. *https://www.blog.google/products/search/how-google-autocomplete-works-search/*, 2018.

Alle URLs wurden zuletzt am 13.01.2020 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Datum und Unterschrift:

Declaration

I hereby declare that the work presented in this thesis is entirely my own. I did not use any other sources and references that the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Date and Signature: