



Logo 1

Logo2

Beuth Hochschule für Technik

Fachbereich 6

# Bachelorarbeit

im Studiengang Medieninformatik - Schwerpunkt Medieninformatik

zur Erlangung des akademischen Grades  
Bachelor of Science

**Thema:** Empfehlungssystem für Kompetenzen

**Autor:** Valentin Risch s55698@beuth-hochschule.de  
MatNr. 798906

**Version vom:** 9. August 2017

**1. Betreuer:** Prof. Dr. Johnanes Konert

**2. Gutachter:** Prof. Dr. Y

## **Zusammenfassung**

## **Abstract**

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>5</b>
<b>Tabellenverzeichnis</b>	<b>6</b>
<b>Listingverzeichnis</b>	<b>7</b>
<b>Abkürzungsverzeichnis</b>	<b>8</b>
<b>1 Einleitung</b>	<b>10</b>
1.1 Inhalte dieser Arbeit . . . . .	10
<b>2 Grundlagen und Begrifflichkeiten</b>	<b>12</b>
2.1 Kompetenzen . . . . .	12
2.2 Kompetenzframeworks . . . . .	12
2.3 RDF, Semantic Web Linked Data . . . . .	13
2.3.1 Semantic Web . . . . .	13
2.3.2 Linked Data . . . . .	13
2.3.3 RDF . . . . .	13
2.3.4 Ontologie/Vokabular . . . . .	14
2.4 Graphen . . . . .	14
2.4.1 Graphentheorie . . . . .	14
2.4.2 Graphendatenbanken . . . . .	15
2.4.3 Neo4J . . . . .	15
2.5 Empfehlungssysteme . . . . .	16
<b>3 Verwandte Themen und Arbeiten</b>	<b>17</b>
3.1 Openbadge . . . . .	17
3.2 Europäischer Qualifikationsrahmen . . . . .	17
3.3 European e-Competence Framework . . . . .	18
3.4 InLoc . . . . .	18
3.5 ESCO . . . . .	18
3.5.1 EURES . . . . .	19
3.5.2 GraphGist: Recommendation System Sandbox . . . . .	19
3.6 RDF Import Neo4J . . . . .	19
3.7 Recommender Systeme . . . . .	19
3.7.1 GraphGist Recommendation Engine Neo4j . . . . .	20
3.7.2 RecSys Challenge . . . . .	20
3.7.3 Job-Matching: Xing . . . . .	20
<b>4 Analyse und Aufgabenstellung</b>	<b>22</b>
4.1 Aufgabenstellung . . . . .	22
4.2 Anforderungsanalyse . . . . .	22
4.2.1 Kompetenzen . . . . .	22
4.2.2 Anforderungen an die Engine . . . . .	23
4.2.3 Anforderungen an die API . . . . .	23
4.2.4 Anforderungen an die Architektur . . . . .	24
4.3 Algorithmen und Metriken . . . . .	24

<b>5</b>	<b>Entwurf</b>	<b>26</b>
5.1	Format/Kompetenzmodell . . . . .	26
5.1.1	ESCO als RDF . . . . .	26
5.1.2	inLOC . . . . .	27
5.1.3	Fazit: Auswahl eines Models . . . . .	27
5.1.4	Datenbank . . . . .	27
5.2	Datenmodell . . . . .	28
5.2.1	API . . . . .	28
5.3	Anfragen Formulieren . . . . .	29
<b>6</b>	<b>Implementierung</b>	<b>30</b>
6.1	Architektur . . . . .	30
6.2	EC2 . . . . .	30
6.3	Neo4j . . . . .	30
6.3.1	RDF Import . . . . .	31
6.4	API Gateway . . . . .	31
6.5	Serverless . . . . .	31
6.5.1	AWS Lambda . . . . .	32
6.5.2	Serverless Framework . . . . .	32
6.6	Architektur im Überblick . . . . .	32
6.7	Deployment mit dem Serverless Framework . . . . .	34
6.8	Algorithmen und Metriken . . . . .	34
6.8.1	Shortest Path . . . . .	35
6.8.2	Jaccard Index . . . . .	35
6.8.3	Levensthein oder Edit-Distanz . . . . .	36

## Abbildungsverzeichnis

1	Beschreibung . . . . .	20
2	Beschreibung . . . . .	24
3	Beschreibung . . . . .	28
4	Beschreibung . . . . .	29
5	Beschreibung . . . . .	32

## **Tabellenverzeichnis**

## Listingverzeichnis

1	Das Listing zeigt einen Funktionsaufruf über die Neo4j . . . . .	19
2	importRDF Prozeduraufruf . . . . .	31
3	Lambda Handler Funktion . . . . .	33
4	Serverless Konfiguration . . . . .	34



## **Abkürzungsverzeichnis**

LOD ..... Linked Open Data

# 1 Einleitung

Mit dem Abschluss einer jeden Berufsausbildung oder eines Studiums beginnt auch der Einstieg in das Berufsleben. Für Viele Bewerber bedeutet das, in Frage kommende Jobangebote zu sichten, und vergleichen. Es folgt das schreiben unzähliger individueller Bewerbungen, denen im schlimmsten Fall nur wenige zu einem Vorstellungsgespräch führen. Doch oftmals kommen schon bereits nach den ersten Wochen des Hochgefühls endlich eine Stelle zu haben, erste Zweifel, ob der Job der richtige für einen ist, und die eigenen Wünsche und Anforderungen erfüllen kann. Absatz

Auch für Personaler stellt die Einstellung von neuem Personal eine schwierige Aufgabe dar. So müssen oft aus einer Reihe von Bewerbern die tatsächlich unqualifizierten aussortiert werden und eine Auswahl getroffen werden, welche Bewerber zu einem Gespräch eingeladen werden oder in die nächste Bewerbungsrunde gelangen.

Es besteht also Bedarf den Prozess von Arbeitsvermittlungen zu optimieren und zu vereinfachen. Das spart sowohl Bewerbern als auch Arbeitgebern viel Frust und Zeit. Ein automatischer Abgleich von Kompetenzen kann diesen Prozess beschleunigen, setzt aber einen einheitlichen Rahmen für die Beschreibung von Kompetenzen voraus. Absatz

Das Kernproblem bei der Arbeitsvermittlung besteht darin, eine vakante Stelle mit einem geeigneten Bewerber zu besetzen. Auf der einen Seite steht also der Bewerber der Kompetenzen anbietet, auf der anderen Seite der Arbeitnehmer, welcher gewisse Kompetenzen fordert. Das technische Problem welches hier umgesetzt werden muss ist, beide Seiten zusammen zu bringen. Der naive Ansatz wäre einfach die Liste der Kompetenzen des Stellenangebotes mit der Liste aus dem Lebenslauf des Bewerbers zu vergleichen und die Passgenauigkeit zu berechnen. Das führt aber unter Umständen dazu, dass interessante Bewerber aussortiert werden weil eventuell andere Bezeichnungen für ihre Fertigkeiten angegeben haben.

Diese Problem wirft die Frage auf welche Standards und Richtlinien gibt es, um Kompetenzen abzubilden und zu modellieren und ob Quellen existieren um Ein Computer basiertes System zu entwickeln welches dies Kompetenzen mit einander vergleicht und Ähnlichkeiten findet. Absatz

Für diese Vergleiche soll ein semantischer Ansatz gefunden werden, es sollen also keine Textvergleiche statt finden, sondern Kompetenzen sollen als Graph abgebildet sein. Diese Arbeit soll sich mit Graphen Basierten Technologien und Ansätzen auseinandersetzen, um Kompetenzen zu modellieren und Ähnlichkeiten zu finden.

## 1.1 Inhalte dieser Arbeit

Der Begriff der *Kompetenz* wird in den Domänen Beruf und Bildung häufig verwendet, oft aber mit unterschiedlicher Definition und Bedeutung. So taucht der Begriff auch häufig in Verbindung mit anderen Substantiven auf. Beispiele sind: Fachkompetenz,


Kernkompetenz, Handlungskompetenz, Teilkompetenz, Selbst- und Sozialkompetenz. Um eine Abgrenzung zu schaffen soll zunächst der Rahmen für den Kompetenzbegriff in dieser Arbeit gespannt werden. **Absatz**

Ein großes wissenschaftliches Interesse besteht im Bereich sogenannter *Recommender Systeme*. Die *Association for Computing Machinery* hält jährlich eine eigene Konferenz zum Thema ab auf der, neustens wissenschaftliche Ergebnisse diskutiert werden. Ein solcher Empfehlungsdienst für die Domäne von Berufs Kompetenzen im Rahmen dieser Arbeit entworfen werden.

Die Arbeit soll die Möglichkeit **evaluieren ob** Kompetenzen als Graph modelliert werden **können** also entweder eine hierarchische Baumstruktur gibt oder die Daten miteinander verknüpft sind. Darum soll eine Technologie ausgewählt um Kompetenzen in einer Graphen Struktur zu speichern, und mit Hilfe dieses Graphen Methoden oder Algorithmen gefunden **werden die** für diesen Anwendungsfall geeignet **sind um** ein Empfehlungssystem zu implementieren.

## Grundlagen und Begrifflichkeiten

### 2.1 Kompetenzen

Im Allgemeinen spricht man bei der Verbindung von Wissen und Können um geforderte Handlungen zu bewältigen von Kompetenzen. Dabei liegt der Fokus auf Fähigkeiten und Wissen, welche dazu beitragen Probleme mit nicht standardmäßigen Handeln zu lösen, und auf verschiedene Situationen zu übertragen. 

Oftmals werden die beiden Begriffe Kompetenz und Qualifikation als Synonyme verwendet, jedoch gibt es signifikante Unterschiede in deren Bedeutung.

**Kompetenz** (lateinisch: competere, zu etwas fähig sein) meint Lernerfolg im Hinblick auf den Lernenden selbst und seine Befähigung zu selbstverantwortlichem Handeln im privaten, beruflichen und gesellschaftlichen Bereich. Sie bezeichnet die subjektive Leistungsfähigkeit einer Person, welche nicht überprüfbar und objektiv bewertbar ist.

**Qualifikationen** (lateinisch: qualis facere, Beschaffenheit herstellen) sind hingegen prüfbar und zertifizierbar. Sie sind die äußere Seite der Leistungsanforderung und auf die Erfüllung vorgegebener Zwecke gerichtet.


[?]

Fachkompetenzen Beispiele für Kompetenzen und Qualifikationen:

Learnin  
Outco-  
mes

### 2.2 Kompetenzframeworks

Kompetenzen abzubilden gestaltet sich als schwierig, da diese einen dynamischen Charakter hat um vom Kontext, ihrer Domäne abhängig ist. Durch die Unterteilung in Branchen und Sektoren ist eine Klassifizierung von Kompetenzen dann möglich. Eine weitere Dimension zur Klassifikation ist das Niveau, es beschreibt das Maß oder auch den "Schwierigkeitsgrad" einer Handlung, die mit der gegebenen Kompetenz zu bewältigen ist. Da Abschlüsse nicht Kompetenz bescheinigen werden Qualifikationsrahmen

benötigt, mit deren Hilfe die in einem Bildungsweg erworbenen Kenntnisse und Fähigkeiten zu Kompetenzentwicklung beitragen können. Mit diesen Lernergebnissen  das "Können" im Sinne von "in der Lage sein, etwas zu tun" beschreiben, ergeben sich dann Vergleichsmöglichkeiten.

Um Qualifikationen und Kompetenzen aus unterschiedlichen sektoralen oder nationalen Rahmenwerken zu vergleichen benötigt es einen Meta-Rahmen. **Absatz**

Zusammenfassung:

"Kompetenzrahmen sind eine wichtige Brücke zwischen dynamischer Kompetenz und statischen Bildungs- und Zertifikatssystemen. Denn Kompetenzrahmen machen Kompetenz klassifizierbar und unterstützen ihre Anerkennung und Anrechnung."

## 2.2 RDF, Semantic Web Linked Data

### 2.2.1 Semantic Web

Das Semantic Web stellt Daten im World Wide Web in einem für Maschinen verarbeitbaren Art und Weise zur Verfügung mit dem Ziel der Interoperabilität, also der Möglichkeit Informationen zwischen Anwendungen und Plattformen auszutauschen und diese in Beziehung zu setzen. Kernaspekte sind das Auffinden relevanter Informationen, die Integration von Informationen aus verschiedenen Quellen und automatische Schlussfolgerung. “Finde Wege und Methoden, Informationen so zu repräsentieren, dass Maschinen damit in einer Art und Weise umgehen können, die aus menschlicher Sicht nützlich und sinnvoll erscheint.”[?, 12]

Im folgenden sollen einige Konzepte und Technologien erläutert werden welche die Anforderungen des Semantic Webs umsetzen.

### 2.2.2 Linked Data

Die Beziehungen der Daten im Semantic Web müssen nicht auf ein Datensatz beschränkt bleiben. So können Daten auch Querverweise auf andere Datensätze haben. Ein Beispiel sind die Daten von Wikipedia die durch das Linked Dataset DBpedia zugänglich sind. Einzelne Einträge beinhalten dabei Verweise auf das Geonames Dataset.

Einige dieser Datensätze sind öffentlich zugänglich (Linked Open Data). So stellt das Amt für Veröffentlichungen der Europäischen Union Europäische Union mit ihrem offenen Datenportal Daten ihrer Institutionen und anderer Einrichtungen zur Verfügung.

**Absatz** Die Vorgehensweise bei der Erstellung von Linked Data wird von Tim Berners-Lee in folgenden 4 Schritten beschrieben:

1. Dinge und Objekte werden durch URIs identifiziert
2. Die URIs sind über das HTTP Protokoll aufrufbar
3. Beim Aufruf werden relevante Informationen in standardisierten Formaten geliefert
4. Die gelieferten Daten enthalten Referenzen auf andere URIs



### 2.2.3 RDF

Um die Anforderung des Semantic Web's, Daten im Web auszutauschen, zu erfüllen, wurde das Resource Description Framework RDF als eine formale Sprache für die Beschreibung strukturierter Informationen geschaffen. Dabei soll die ursprüngliche Bedeutung erhalten bleiben und Kombinationen und Weiterbearbeitung der enthaltenen Informationen ermöglicht werden. Eine Resource kann generell jedes Objekt

mit einer eindeutigen Identität sein. Zb. Bücher, Orte, Menschen, abstrakte Konzepte usw. Um Mehrdeutigkeiten zu vermeiden werden URIs als Bezeichner verwendet. Diese RDF-Beschreibungen können auch durch Zeichenketten syntaktisch dargestellt werden, müssen vorher jedoch in Bestandteile zerlegt und serialisiert werden. Die dabei entstehenden Dokumente sind gerichtete Graphen, wobei Knoten und Kanten mit eindeutigen Bezeichnern beschriftet sind.

Triple RDF-Graphen lassen sich vollständig durch ihre Kanten beschreiben. Eine solche Kante hat einen Anfangspunkt, eine Beschriftung und einen Endpunkt. Dieses Triple wird bestimmt durch "Subjekt-Prädikat-Objekt".

## 2.2.4 Ontologie/Vokabular

Als Ontologie oder Vokabular wird eine Sammlung von Begriffen bezeichnet, die innerhalb einer Domäne Wissen abbilden und klassifiziert werden können. [?] Mit Hilfe eines Vokabulars werden Regeln über die Semantik von Klassen, Attributen und Beziehungen von Daten definiert. Ontologien können auch in einer spezifischeren Domäne erweitert werden. In der *Simple Knowledge Organization System* Ontologie wird durch die *Concept* Klasse eine Idee oder ein Sachverhalt abgebildet und mit anderen verknüpft. Auch Vererbung und hierarchische Strukturen sind möglich.

Vokabu-  
bzw.  
Ontolo-  
gien

## 2.4 Graphen

Ein Graph ist definiert als eine Menge von Knoten (Vertices) und deren Beziehungen, welche über Kanten (Edges) dargestellt werden. Mit Graphen kann man vernetzte Strukturen wie zb. Straßennetze, Computernetzwerke oder Datenstrukturen modellieren. So finden sich in vielen modernen Technologien wie Routenplaner oder Social Media Anwendungen graphentheoretische Konzepte wieder.

Ontolog-  
ausar-  
beiten

### 2.4.1 Graphentheorie

In diesem Teilgebiet der Mathematik werden Graphen und ihre Beziehungen zueinander untersucht. Das älteste dokumentierte Problem der Graphentheorie ist das Königsberger Brückenproblem. Dabei wurde ein Rundweg durch Königsberg gesucht, der alle Brücken jedoch nur einmal überquerte. Leonhard Euler erkannte 1736, dass man die einzelnen Ufer als Punkte und Brücken als Kanten abstrahieren konnte. Euler zeigte, dass ein solcher Weg nicht existierte, da jeder Knoten mit einer ungeraden Anzahl von ungerichteten Kanten verbunden sein muss. **Absatz**

Da dieses Themengebiet sehr umfassend ist sollen hier nur einige für diese Arbeit relevante Konzepte erwähnt werden.

**Gerichteter Graph:**

Ein gerichteter Graph ist definiert als  $G = (V, R, \alpha, \omega)$  mit  $V$  als nicht leere Menge von Knoten,  $R$  als Kantenmenge.  $\alpha$  und  $\omega$  sind jeweils Abbildungen für die gilt  $r(\alpha, \omega)$ , wobei  $r$  eine Kante ist die ihren Ursprung im Anfangsknoten  $\alpha$  hat und im Endknoten  $\omega$  endet. **Absatz**

**Adjazenzmatrix:**

Die Adjazenzmatrix oder auch Nachbarschaftsmatrix ist eine  $n \times n$  Matrix mit  $n = |V|$ . Sie gibt welche Knoten im Graphen miteinander verbunden sind. Für die Adjazenzmatrix  $A(G)$  gilt  $a_{ij} = |\{ r \in R : \alpha(r) = v_i \text{ und } \omega(r) = v_j \}|$ . **Absatz**

**Gewichtete Kanten:**

Eine Kante kann bewertet werden. Sie wird mit einer Zahl notiert. Diese Zahl kann dann als Parameter verwendet werden und gibt die Kosten an, die anfallen wenn die Kante von einem Algorithmus oder einer Funktion passiert wird.

graphentheoretische Konzepte zitieren Seite4

**2.1.2 Graphendatenbanken**

Im Gegensatz zu relationalen Datenbanken werden Daten nicht in **Tupeln** als Tabellen gespeichert sondern als Knoten in einem Graphenmodell. Der Vorteil darin liegt zunächst in der Daten Modellierung, da das Graphen Model sehr intuitive ist und sich einfach auf Papier modellieren lässt. Ein weiterer wichtiger Aspekt ist der Performance Vorteil bei **Abfragen** von stark verknüpften Daten. Abfragen in relationalen Datenbanken werden **langsamer** je größer der Datenbestand. Im Graphen wird immer nur der Teilgraph **durchlaufen** welcher die die Abfrage erfüllt.[?, 8]


**2.1.3 Neo4J**

Die Open Source Graphdatenbank Neo4j ist durch ein **Labeled Property Graph Model** implementiert. Knoten im Graph können durch Labels in Kategorien eingeordnet werden und speichern Informationen als Schlüssel-Wert Paare in Properties. Neben Knoten können auch Kanten **Properties** erhalten, was eine Gewichtung der Beziehungen zwischen Knoten ermöglicht. Eine Beziehung besteht immer aus einem Start- und Endknoten, sowie einer Richtung.[?, 26]

Anfragen an die Neo4j Datenbank werden mittels der deklarativen Sprache *Cypher* ausgeführt. Mit dieser Sprache können Daten gefunden werden, welche einem Muster entsprechen. Diese Patterns setzen sich aus Knoten und Beziehungen zusammen.

Mit der *MATCH* Klausel, werden alle Pfade im Graphen gezeigt, welche dem Pattern entsprechen. Mit *WHERE* kann die Sammlung gefiltert werden.

## 2.5 Empfehlungssysteme

Ein Empfehlungssystem ist eine Software, welche Nutzern Vorschläge zu Artikeln, Music, Filmen, Büchern oder anderen Objekten macht. [?] Ausgehend von einem Objekt werden dem Benutzer andere Objekte mit Ähnlichkeiten zu dem Ausgangsobjekt aufgelistet. Für die Ergebnisse werden im wesentlichen zwei Ansätze verfolgt. Das Inhaltsbasierte Filter  und Kollaborative Filtern.



## Verwandte Themen und Arbeiten

### 3.1 Openbadge

Auf unserem Bildungsweg werden das Erlangen von Fertigkeiten und Kenntnissen mit Zeugnissen und Abschlüssen belegt. Doch oftmals genügt die formale Ausbildung nicht oder hat aufgrund der sich schnell verändernden Technologien oder Kompetenzen nur eine begrenzte Gültigkeit. Die Europäische Union fordert eine stärkere Anerkennung von informalem Lernen, damit auch Fertigkeiten und Kenntnisse die ohne ein formales Abschlusszertifikat erworben wurden Anerkennung finden.[?]

Doch wie können Personen alle Ihre Fertigkeiten, welche an einer Hochschule, in einer staatlich Anerkannten Ausbildung oder in einem Online Seminar, in einem Workshop etc. erworben wurden präsentieren, damit auch Arbeitgeber und Bildungsinstitute in der Lage sind, sicherzustellen, dass Bewerber die nötigen Fertigkeiten mitbringen.[?]

In der digitalen Welt können sogenannte Badges ein Lösungsansatz sein. Ein digitaler Badge ist ein digitales Zertifikat für eine erbrachte Leistung oder eine Fähigkeit. Die Mozilla Foundation hat in Zusammenarbeit mit der MacArthur Foundation den Open Badge(OB) Standard entwickelt. Er stellt sicher, dass Alle Badges Informationen über Kriterien und Nachweise erhalten. Die Informationen in einem Badge können auch auf ein Kompetenzframework verweisen, und validiert werden.[?, 4]

Badges können von Institutionen, Schulen und Arbeitgebern verliehen werden. Sie definieren ein Set von Kompetenzen oder einen Lehrplan und eine Bewertung um festzustellen ob ein Empfänger die notwendigen Anforderungen erfüllt hat. Darüber hinaus können Badges von ihrem "issuer" mit einer verschlüsselten Zusicherung versehen werden, welche bestätigt, dass der "earner" des Badges die geforderte Leistung auch erbracht hat. Die Zusicherung kann dann in den Quellcode eines SVG oder PNG Bildes geschrieben werden, sodass dritte später eine elektronisch Überprüfung beim Herausgeber beantragen können. Über das Alignment-Attribut kann ein Badge auch auf eine Quelle verweisen, welche die Kompetenz oder Fähigkeit beschreibt.

### 3.2 Europäischer Qualifikationsrahmen

Der EQR ist ein Meta-Rahmen, welcher die Vergleichbarkeit von beruflichen Qualifikationen und Kompetenzen aus verschiedenen nationalen oder sektoralen Kompetenzrahmen ermöglichen soll. vergleichbarkeit von beruflichen Qualifikationen und Kompetenzen

ausführ-  
nut-  
zung  
von  
eqf in  
ESCO

### 3.3 European e-Competence Framework

Der europäische Kompetenzrahmen für Fach- und Führungskräfte der Informations- und Kommunikationstechnologiebranche, ist eine sektor-spezifische Umsetzung des Europäischen Qualifikationsrahmens EQR. Er unterteilt Kompetenzen in 5 Feldern auf 5

Niveaus

### 3.4 InLoc

Das Europäische Projekt InLOC(Integrating Learning Outcomes and Competences) erlaubt es Kompetenzen und Lernergebnisse verschiedener Kompetenzrahmen in einem einheitlichen semantischen Format abzubilden.

### 3.5 ESCO

In der EU gibt es nach einer Aktuellen Eurostat Statistik ca. 19 Millionen Menschen ohne Beschäftigung. Jedoch haben einige Branchen in Deutschland Probleme Stellen mit qualifiziertem Personal zu besetzen. So blieben im Jahr 2016 In der IT und Telekommunikationsbranche 375.034 Stellen unbesetzt.[?]

Die Europäische Kommission hat dieses Problem erkannt und mit ESCO eine mehrsprachige Klassifizierung für europäische Fähigkeiten, Kompetenzen, Qualifikationen und Berufe entwickelt, deren Zusammenhang durch Berufsprofile verdeutlicht wird.

Eine der Aufgaben von ESCO ist es, Die Lücken zwischen dem Arbeitsmarkt und den verschiedenen Bildungssystemen der einzelnen Mitgliedstaaten zu schließen. So unterscheiden sich Qualifikationen, welche Menschen in ihren Heimatländern erhalten nicht nur voneinander, sondern können auch oftmals nicht mit aktuellen Entwicklungen des Arbeitsmarktes und dessen Anforderungen mithalten. Absatz


Die ESCO Daten werden gemäß den Praktiken für Linked Open Data veröffentlicht. Dies soll Entwicklern den Zugriff erleichtern und Anwendungen für Stellenausgleich, Berufsberatung und Selbsteinschätzungen ermöglichen.

Die Europäische Kommission hat mit ESCO eine Schnittstelle geschaffen, welche Informationen zwischen den nationalen Klassifizierung Systemen übersetzen soll und somit eine höhere semantische Interoperabilität schaffen wird. Ein Hauptinteresse von ESCO ist der kompetenzbasierte Job Abgleich. Arbeitnehmer sollen ihre eigenen Fähigkeiten, Kompetenzen und Qualifikationen mit freien Stellen vergleichen können, und so eventuelle Kompetenzlücken identifizieren zu können. Auf der anderen Seite, muss es Arbeitgebern möglich sein, Stellenausschreibungen durch Fähigkeiten, Kompetenzen und Qualifikationen zu beschreiben, und Bewerber mit den geforderten Kompetenzen abzugleichen. Diese Anforderungen müssen nun von IT-System erfüllt werden.

Wichtig  
Mehr  
Infor-  
ma-  
tionen  
ausar-  
beiten

## 1 EURES

### 3.5.2 GraphGist: Recommendation System Sandbox

Neo4J bietet über die **Sand** eine interaktive Möglichkeit auf einer temporär generierten Instanz im Browser zu arbeiten. Mit Schritt-für-Schritt Anleitungen werden Themen wie "Netzwerk Management" oder die "Panama Papers" in Neo4j näher gebracht.

Ein für diese Arbeit relevantes Themenfeld bietet die Sandbox "Recommendations".[?] Als Datenquelle für die Instanz stehen die "Open Movie Database"[?], und das MovieLens Projekt[?] zur Verfügung. Neben einer Erläuterung zum Property Graph Model und einer Einführung in die Cypher Query Language gibt es Beispiele zu verschiedenen Methoden und Metriken für das Filtern von Ergebnissen. Dabei werden verschiedene Ansätze zu den Methoden Inhaltsbasierte Filterung und Kollaborative Filterung erläutert.

## 6 RDF Import Neo4J


Jesús Barrasa ist *Senior Graph Solutions Consultant at Neo Technology* bei Neo4j, und hat ein Plugin für Neo4j entworfen, mit dessen Hilfe sich RDF Dokumente in Neo4j importieren lassen. So lässt sich mit dem Befehl :

```
1 CALL semantics.importRDF("file:///.../esco_skos.rdf", "RDF/XML",
2 { languageFilter: 'de', commitSize: 5000 , nodeCacheSize: 250000})
```

Listing 1: Das Listing zeigt einen Funktionsaufruf über die Neo4j

Der komplette RDF Graph des ESCO Kataloges mit allen Beziehungen laden und Abfragen.

## 3.7 Recommender Systeme

Bekannte Anwendungsfälle für ein Recommender System sind neben Musik und Film Empfehlungen vor allem die **andere Kunden kauften auch** **Komponente** in Web Shops. Ziel dieser Systeme ist es eine Vorhersage zu treffen, welches Produkt oder Objekt dem Kunden oder Anwender ebenfalls interessieren könnte. Dabei werden häufig die Konzepte des *Inhaltsbasierenden* und des *Kollaborativen* Filtern angewandt.

Beim Inhaltsbasierten Filtern wird die Ähnlichkeit der Objekte und deren Eigenschaften ermittelt. Im Gegensatz dazu werden beim kollaborativen Filtern Benutzer und deren Präferenzen miteinander verglichen.

### 3.7.1 GraphGist Recommendation Engine Neo4j

### 3.7.2 RecSys Challenge

Die *ACM RecSys Conference* befasst sich jedes Jahr mit den aktuellen Untersuchungsergebnissen und Techniken im Bereich der Empfehlungsdienste. Darüber hinaus veranstaltet sie auch die RecSys Challenge, einen mit 3000 € **Siegerprämie** dotierten Wettbewerb, bei dem Teilnehmer ein Empfehlungsdienst, in einer bestimmten Domäne entwickeln sollen. In den beiden vergangenen Jahren beschäftigte sich der Wettbewerb mit Job Empfehlungen der Karriereplattform Xing. Die Aufgabe war, anhand eines neuen Job Angebotes, all jene Nutzer zu finden, die interessiert sein könnten eine Benachrichtigung über das neue Angebot zu **erhalten aber** auch Nutzer die ebenfalls für den Job geeignet sein könnten. Die Ergebnisse werden auf der RecSys Conference in Como, Italien am 27.08.2017 vorgestellt.

### 3.7.3 Job-Matching: Xing

Das Karrierenetzwerk bietet Arbeitssuchenden die Möglichkeit Job-Empfehlungen auf Basis des eigenen Profils zu erhalten. Dabei wird nach den Kriterien Entfernung zum Arbeitsort, Karrierestufe, Skills und Aktivitäten gefiltert. Aus diesen 4 Kriterien wird ein Relevanz-Indikator ermittelt, welcher die Treffgenauigkeit des Inserats angibt. Umgekehrt wird auch ein Inserat mit dem eigenen Profil gematcht, wobei auch die geforderten Skills den eigenen gegenübergestellt werden.[?]

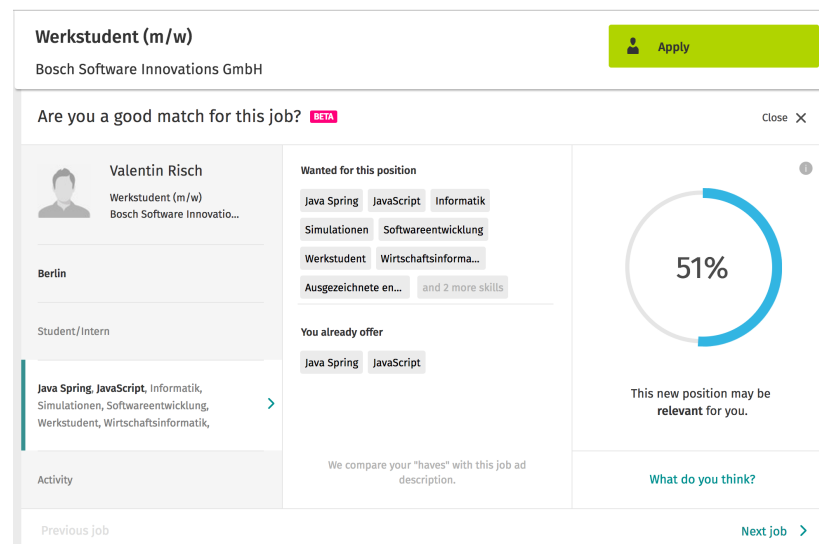


Abbildung 1: Jobmatching Xing-Profil

Dieser Ansatz berücksichtigt jedoch nicht die Persönlichkeit der Bewerber, und bietet auch nur eine teilweise höhere Bewerberpassgenauigkeit aus Sicht der Arbeitgeber. Wie qualifiziert ein Bewerber wirklich ist bleibt weiter unklar, da eine Liste der angegebenen Skills keine Auskunft darüber gibt wie kompetent der Bewerber auf einem Gebiet

wirklich ist. So kann es passieren, dass ein Bewerber der trotz Eignung durch das Raster fällt, weil er andere Schlagworte als die geforderten angegeben hat.

Neben Xing bieten die Plattformen Truffles und BirdieMatch ähnliche Technologien an.

## 4 Analyse und Aufgabenstellung

Eine der großen Herausforderung in der Personalentwicklung ist das Job-Matching, eine vakante Stelle mit einem passenden Bewerber oder Angestellten zu besetzen. Dabei greifen Personaler vermehrt auf Technologien und Matching-Algorithmen zurück, die diesen Prozess automatisieren sollen. Das Prinzip ähnelt dem, der Partnervermittlung. Person A, Der Arbeitgeber wird mit Person B, dem Arbeitssuchenden oder Angestellten zusammen gebracht. Unter Berücksichtigung von Kompetenzen die eine Stelle erfordert, kann dieses Matching Verfahren verfeinert werden.

Im Kontext der Personalentwicklung, Jobvermittlung oder Berufsbildung ist es aber auch wichtig Karrierepfade aufzuzeigen, also heraus zu finden welche Kompetenzen sich ähnlich sind, welche Kompetenzen fehlen um die Anforderungen für eine bestimmte Tätigkeit zu erfüllen. Dieser Prozess kann durch Algorithmen und Metriken implementiert werden.

### 4.1 Aufgabenstellung

Ein wichtiger Bestandteil von Stellenbeschreibungen sind auch eine Liste von geforderten Kompetenzen. Kompetenzen lassen sich als Graph modellieren, denn oftmals bestehen zwischen ihnen hierarchische Strukturen und Gemeinsamkeiten. Das Ziel dieser Arbeit soll es sein, Die Beziehungen dieser Kompetenzen in einem Graphenmodell zu analysieren, und Methoden und Metriken zu finden mit deren Hilfe es möglich sein wird, Ähnlichkeiten von einzelnen Kompetenzen oder ganzen Kompetenzsets zu ermitteln. Konkret soll ein Dienst entwickelt werden, der von anderen Programmen genutzt, oder in bestehende Systeme integriert werden kann.


### 4.2 Anforderungsanalyse

#### 4.2.1 Kompetenzen

Kompetenzen sind die Entitäten welche von dem Empfehlungssystem verarbeitet werden sollen. Sie können ihren Ursprung außerhalb des Systems haben, werden aber innerhalb in einer Datenschicht persistiert. Ursprung von Kompetenzen können ein *Kompetenzrepository* wie es Herr Lopez in seiner Arbeit *Entwicklung und Evaluation eines e-Kompetenz-Verzeichnisses mit REST-API und eines automatisierten Crawlers zur Datensammlung*[?] entwickelt hat, oder ein Katalog mit Kompetenzen wie *ESCO* von der Europäischen Kommission. Auch Kompetenzen aus Kompetenzrahmen wären als Quelle denkbar, sofern diese in einem maschinenlesbaren Format zur Verfügung stehen. Eine weitere denkbare Möglichkeit wäre das auslesen des *criteria* oder *alignment*

Feldes von Open Badges, wie  in der Veröffentlichung *proposal on Competency Alignment and Directory* [?] des Open Badge Networks vorgeschlagen wird.

## 2.2 Anforderungen an die Engine

Ein solches Empfehlungssystem soll als eigener Dienst fungieren, und über eine API erreichbar sein. Die Eingabe für das Empfehlungssystem soll eine Kompetenz oder eine ein Set von Kompetenzen sein. Kompetenzen werden in einer Datenschicht persistiert, die die gängigen CRUD Datenbankoperationen ermöglicht. Die API leitet die Kompetenzen dann an eine Methode weiter, welche die nötigen Entitäten und deren Beziehungen zurück liefert. Auf resultierenden Datensätzen können weitere Operationen, wie Aggregation oder Sortierung durchgeführt werden und falls weitere Berechnungen oder Operationen nötig sind die mit der Abfragensprache nicht geleistet werden, mit einer  gängigen Programmiersprache weitere algorithmische Schritte implementieren. Am Ende sollen 0, 1 oder mehrere Kompetenzen in sortierter Reihenfolge an den Aufrufer zurückgeliefert werden. **Absatz**

Folgende Anforderungen können an dieses System gestellt werden.

- Übereinstimmung von Kompetenzsets ermitteln
- Ähnlichkeit von Kompetenzsets ermitteln
- Ähnliche Kompetenzen finden
- Inkludierende Kompetenzen ermitteln
- Fehlende Kompetenzen ermitteln



### **Absatz**

Optionale Anforderungen

- Vergleich von Kompetenzen aus verschiedenen Frameworks

## 2.3 Anforderungen an die API

Das Empfehlungssystem soll über eine API für andere Softwaresysteme verfügbar sein. Beispiele für solche Systeme können Jobportale oder Human Resource Management Systeme sein. Sie können über HTTP Requests Daten mit Kompetenzsets an die Engine senden und erhalten verarbeitete Daten als Antwort.

Mit der Übereinstimmung von Kompetenzsets   das einer Stellenbeschreibung und das eines Lebenslaufes, kann ein Matching vorgenommen werden, welches Computersystemen ermöglicht passende Stellenangebote oder Bewerber zu finden. Über die Vernetzung der Kompetenzen im Graphen können Wege zur weiteren Bildung ermittelt werden.

#### 4 Anforderungen an die Architektur

Der Dienst soll nach dem Microservice Muster implementiert werden, also bei Bedarf komplett austauschbar sein und wenige Abhängigkeiten enthalten. Nutzer sollen über das HTTP Protokoll mit dem Dienst kommunizieren können. Die Daten werden in einem Graphen persistiert, in welchem Kompetenzen als Knoten und Beziehungen der Kompetenzen als Kanten gespeichert werden. Für die Kommunikation soll ein gängiges Format zur Ein und Ausgabe gewählt werden.

- Erreichbarkeit über eine Web-API
- Persistenz der Daten in einem Graphen
- Daten Ein und Ausgabe in einem standardisierten Format
- Skalierbarkeit: soll viele Anfragen gleichzeitig bearbeiten können

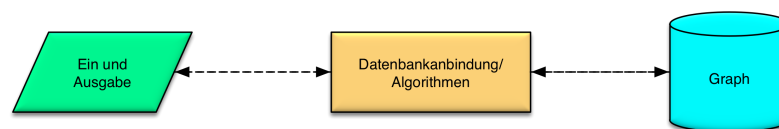


Abbildung 2: Entwurf

#### 5 Algorithmen und Metriken

Das Kernproblem des Empfehlungsdienstes ist die Berechnung von Ähnlichkeiten von Knoten in einem Graphen. Es gibt bereits einige Algorithmen und Metriken die sich mit dieser Problematik auseinander setzen. Welche Metriken und Algorithmen angewandt werden können hängt aber auch vom Anwendungsfall ab. Einige Methoden zur Berechnung von Ähnlichkeiten sollen im folgenden aufgelistet, und später bei der Implementierung evaluiert werden. Ein grundsätzlicher Ansatz kann lautet, *je näher zwei Knoten im Graphen einander sind, desto ähnlicher sind sie sich*. Der folgende Graph soll dies veranschaulichen. **Absatz**

Das Problem des *kürzesten Pfades* ist ein häufig behandeltes Problem in der Graphentheorie. Ein kürzester Pfad von einem Knoten  $s$  zu  $t$  in einem Graphen  $G$  ist definiert durch eine Funktion  $c: R \rightarrow \mathbb{R}$

Es gibt aber auch mehr Eigenschaften des Graphen die sich ein Algorithmus zu nutzen machen kann. So können mit Hilfe von gewichteten Kanten Vektoren entlang der Pfade von einem Knoten zu einem anderen erstellt werden. Die *Kosinus-Ähnlichkeit* ist eine Metrik den eingeschlossenen Winkel zwischen zwei solcher Vektoren ermittelt.



Mit dem **Jaccard Koeffizienten** lassen sich zwei Mengen auf ihre Ähnlichkeit prüfen. Der Wert des Koeffizienten ist genau dann 1, wenn die beiden Mengen identisch sind. Je weniger Gemeinsame Elemente zwei Mengen besitzen, desto kleiner wird der Koeffizient und wird 0, wenn keine Gemeinsamkeiten vorliegen.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

In einem Graphen kann die Metrik für 2 Knoten angewandt werden, indem man die beiden Mengen aus Knoten zusammen setzt, die eine Beziehung zu den 2 Knoten haben. Kategorien oder Taxonomien sind dabei geeignete Kandidaten für gemeinsame Beziehungen. Am Beispiel des Movie Graphen, kann hier eine Ähnlichkeit von 2 Filmen berechnet werden, die sich in gemeinsamen Genres befinden. Je mehr gemeinsame Genres, desto höher die Ähnlichkeit. Sind beide Filme in denselben Genres zb. {Horror, Action, Comedy} auf, haben sie eine Ähnlichkeit von 1, haben sie kein gemeinsames Genre ist die Ähnlichkeit 0.

Ein weiterer Ansatz lautet *"Zwei Objekte sind Ähnlich, wenn sie mit ähnlichen Objekten verknüpft sind"*. Der **SimRank** [?] Algorithmus versucht Ähnlichkeiten von Knoten zu finden zwischen denen ein struktureller Zusammenhang besteht.

Ontologien im Semantic Web haben eine Baumstruktur. Um semantische Nähe zwischen Konzepten zu berechnen müssen neben der Distanz der Knoten auch die Tiefe im Baum berücksichtigt werden. **Wu und Palmer** [?] schlagen folgende Berechnung für 2 verwandte Konzepte  $C_1$  und  $C_2$  vor:

$$Sim_{wp} = \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3}$$

Wobei  $N_3$  die Länge des Pfades vom Wurzelknoten bis zum nächsten gemeinsamen Knoten  $C_3$ ,  $N_2$  die Länge des Pfades  $p(C_3, C_2)$  und  $N_1$  die Länge des Pfades  $p'(C_1, C_3)$  ist.

Mit der **Edit-Distanz** [?] wird mit Hilfe von Edit-Operationen, also dem löschen und hinzufügen von Kanten eine Graphentransformation durchgeführt, und für jede Edit-Operation  $\eta$  Transformationskosten  $c(\eta)$  veranschlagt. Die Summe der minimalen Kosten der Transformationsfolge  $D$  kann dann als Maß für eine Ähnlichkeit von zwei Graphen  $G_1$  und  $G_2$  verwendet werden.

## 5 Entwurf

Im folgenden sollen die aus der Analysephase ausgearbeiteten Anforderungen zu einem Entwurf zusammen gefasst werden, in dem die notwendigen Komponenten und deren Schnittstellen genauer beschrieben werden. Zunächst soll aber das Datenmodell für die Kompetenzen und dessen Format gewählt werden.

### 5.1 Format/Kompetenzmodell

Wie in Kapitel 1 beschrieben, gibt es unterschiedliche Ansätze und widersprüchliche Meinungen um ein Modell für Kompetenzen zu beschreiben. Für den Entwurf des Empfehlungssystems sollen zunächst einige in Frage kommenden Modelle evaluiert werden.

#### 5.1.1 ESCO als RDF

ESCO wird als RDF Graph im SKOS Format zum Download angeboten. Das Vokabular für dieses Schema beinhaltet unter anderem folgende relevante Klassen: [Absatz](#)

[ESCO concepts Relationship](#)

IRI: <http://data.europa.eu/esco/modelRelationship> [Absatz](#)

Beziehung zwischen zwei ESCO Säulen zb. Occupation und Qualification.

[Qualification](#)

IRI: <http://data.europa.eu/esco/modelQualification> [Absatz](#)

Ein offiziell oder formelles Zertifikat eines oder mehrerer Kompetenzen oder Fertigkeiten. [Absatz](#)

Kann verknüpft mit Skills oder Berufen sein.

[Skill](#)

IRI: <http://data.europa.eu/esco/modelSkill>

In dieser Klasse werden Konzepte für Fertigkeiten und Kompetenzen und Wissen zusammengefasst. [Absatz](#) tendet hier dieselben Definitionen für diese Begriffe an wie sie der EQF vorgibt.[?] ESCO unterscheidet in seinem Modell nicht zwischen Fertigkeiten und Kompetenzen sondern nur zwischen Fertigkeiten und Wissen. Dieser Unterschied wird durch hierarchische Strukturen implementiert. [Absatz](#)

[Occupation](#)

IRI: <http://data.europa.eu/esco/modelOccupation>

In dieser Klasse werden Berufsgruppen zusammengefasst. [Absatz](#)

Kritik für das ESCO Modell kommt hingegen vom Bundes Ministerium für Berufliche Bildung. Es sieht mögliche negative Rückwirkungen auf duale Berufsbildungssysteme, beim Einsatz von europaweiten Teilkompetenzen und fordert eine genauere Überprüfung der Kompatibilität von ESCO mit Standards wie EQR und NQR hinsichtlich der Qualifikationsniveaus der Beschreibungen.

## 2 inLOC

Das Projekt InLOC ermöglicht das Abbilden von mehreren Kompetenzrahmen in einem einheitlichen maschinenlesbaren Format(RDF,XML,JSON-LD).




Auf der Projekt Website wird anhand einer exemplarischen Kompetenz des e-CF in Version 2.0 die Struktur des inLOC Informations Modells beschrieben.

Das folgende Digram zeigt die Struktur für Kompetenzen aus dem e-CF. Deutlich zu erkennen sind die 4 Dimensionen *Kompetenzfelder*, *Kompetenzen*, *Kompetenzniveaus*, *Beispiele für Wissen und Fähigkeiten*



## 3 Fazit: Auswahl eines Modells

Das inLOC Model findet aktuell keine Anwendung, und man müsste eine erst eine Reihe von Beispiel Daten erstellen, zb. durch manuelles eintragen der Kompetenzen aus dem e-CF.

Für die Implementierung soll zunächst nur das Modell  welches durch den ESCO Katalog angeboten wird verwendet werden. Zum einen Stellt dieser Katalog in der aktuellen Version mit ca. 13.500 Einträgen zu Wissens und Fertigkeiten/Kompetenz Konzepten zum Download zur Verfügung, und ist darüber hinaus EU Länder übergreifend  was ihn für einen breiteres Spektrum an Anwendungen interessant macht. Der Aspekt  der Kompatibilität mit einzelnen nationalen Bildungssystemen kann vernachlässigt werden, da es nicht um die Findung einer Lösung geht, welche den Arbeitsmarkt und die Berufsbildung in gleichem Maße berücksichtigt, sondern den rein technischen Ansatz, für ein Verfahren Kompetenzen aufgrund ihrer strukturellen Modellierung zu vergleichen.

## 4 Datenbank

Nachdem nun das Datenmodell für die Speicherung der Kompetenzen gewählt wurde, muss eine geeignete Datenbank gewählt werden. Das SKOS Schema des ESCO Katalogs liegt zwar bereits als gerichteter Graph, nämlich im RDF Format vor, soll aber aus Gründen der Flexibilität und Möglichkeiten in ein Datenbankmanagement System überführt werden.

Alternativ könnte mit Hilfe von SPARQL, einer Graphen basierten Abfragesprache, direkt auf dem RDF Graphen gearbeitet werden ,hier sollen jedoch von den Vorteilen eines *Label Property Graphen* Gebrauch gemacht werden. Möchte man eine Kompetenz in RDF modellieren, so muss zunächst ein Knoten für die Kompetenz selbst angelegt werden. Soll die Kompetenz neben dem *Identifizier* auch noch einen Namens oder Sprachattribut hinzufügen, müssen zunächst Knoten und Beziehungen für jedes Attribut angelegt werden. Daraus resultiert ein stärkere Vernetzung des Graphen. Im *LPG* können die Attribute einfach dem Kompetenzknoten hinzugefügt werden. Emil Eifrem, CEO der Open-Source Graphendatenbank [?] übt Kritik an der RDF Community. So

bemängelt er, dass zu wenig Rücksicht auf Entwickler und die Integration in Software Systeme genommen wird. [?]

"There is a pool of super smart people in the Semantic Web community, but their approach is typically extremely academic. " Warum keine RDF Technologie?

## 5.2 Datenmodell

Da Der ESCO Katalog als Ontologie implementiert und im Linked Data Format als SKOS veröffentlicht wird, liegen die Konzepte von Kompetenzen als hierarchische Struktur vor. Dabei gibt es ausgehend vom Root Knoten einen Unterknoten für die *Kompetenzen/Fertigkeiten* Säule, von dem ausgehend sich der Baum aufspaltet in Job-spezifische und Transversale Skills. Hierarchien in SKOS werden über *broader-narrower* Beziehungen hergestellt. Kompetenzen sind Teil der Konzept Klasse *Skill* und werden mit der *LeafGroup* und *Member* Konzept Klasse strukturiert. Beispielsweise ist die Kompetenz *Windows XP* Teil der Gruppe *Computing* die wiederum Teil der Gruppe *Job-specific skills/competences* Gruppe ist.

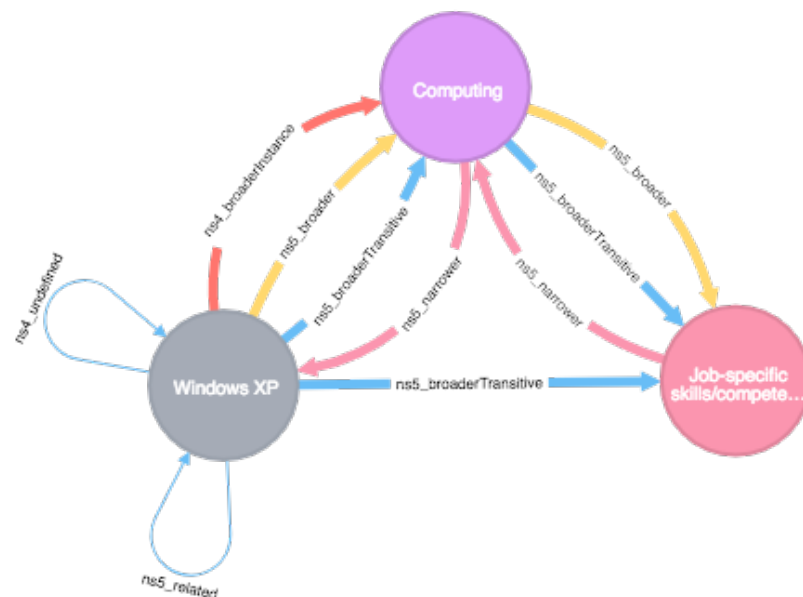


Abbildung 3: Hierarchische Struktur von Kompetenzen

### 5.2.1 API

Die Funktionsweise der Recommendation Engine soll sich lediglich auf folgende Prozesse beschränken:

Eingabe und Ausgabe einer Kompetenz Berechnung von Ähnlichkeiten im Graphen durch Graph Traversierung und Abfragen an die Datenbank.

Über das HTTP Protokoll sollen Anwendungen mit dem Empfehlungsdienst kommunizieren können. Dabei werden die CRUD Methoden POST und GET auf entsprechenden Endpunkten der aufgerufen. Absatz

Die Zweite Komponente muss eine Datenbankanbindung für die Datenbank gewährleisten, und Daten aus der Eingabe in Anfragen an die Datenbank verarbeiten, sowie Ergebnisse der Anfragen zurückliefern. Falls die Anfragensprache der Datenbank nicht alle Möglichkeiten ausschöpfen kann die Ergebnisse wie gewünscht zu filtern, können in der zweiten Komponente weitere Algorithmen implementiert werden.

Die Anwendung ist also wenig komplex. Ein viel höherer Implementierungsaufwand ist den Vergleichsalgorithmen, bzw Graph Queries zuzurechnen.

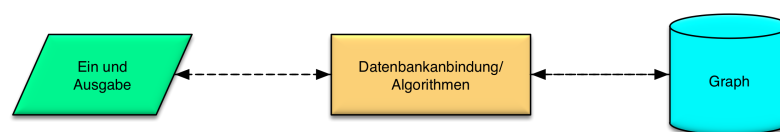


Abbildung 4: Entwurf



## Anfragen Formulieren

Bevor Abfragen in einer Abfragensprache für die Datenbank erstellt werden, sollen diese zunächst in natürlicher Sprache formuliert werden, und später in eine entsprechende Abfragensprache transformiert werden. Dabei soll das zuvor erstellte Datenmodell helfen. Anhand der Knoten und Kanten können die Entitäten und deren Beziehungen identifiziert werden, aus diesen lassen sich dann Abfrage Sätze formulieren. In der Formulierung wird zum Ausdruck gebracht welche die gewünschten Knoten sind und welcher Bedingung das Ergebnis unterliegt. Die Einfachste Abfrage würde demnach lauten. *Zeige Alle Knoten.* Das Ergebnis wäre der Komplette Graph. Eine spezifischere Ergebnis erhält man durch das hinzufügen von Bedingungen. *Zeige Alle Knoten, mit dem Attribut name={Softwareentwicklung}.*

## Implementierung

In diesem Kapitel sollen die einzelnen Implementierungsschritte für den Empfehlungsdienst erläutert werden. Alle Komponenten können in der *Amazon Webservices Cloud* implementiert werden. Dabei sollen nur die eingesetzten Dienste beschrieben werden, jedoch auf grundlegende Einstellungen wie die Benutzer Rollen und Berechtigungen in der AWS Konsole über welche sich sämtliche Dienste steuern und konfigurieren lassen, nicht weiter eingegangen werden. Im einzelnen kommen folgende Dienste zum Einsatz:

### 1 Architektur

#### EC2 [Absatz](#)

Mit Elastic Compute Cloud können virtuelle Rechner Umgebungen erstellt werden auf welchen skalierbare Applikationen entwickelt werden können. Rechen und Speicherkapazitäten können den persönlichen Bedürfnissen angepasst werden. [Absatz](#)

AWS API Gateway: [Absatz](#)

Mit API Gateway können Anwendungen über das Internet auf die Cloud Dienste von Amazon zugreifen. Dazu werden HTTP-Endpunkte erstellt. [Absatz](#)

AWS Lambda [Absatz](#)

Code in Lambda Funktionen wird nur ausgeführt wenn entsprechende Events die Lambda Funktion auslösen.

### EC2

Für die Datenhaltung wird das Open Source Graphendatenbank Manangement System Neo4j verwendet. Dieses stellt jedoch einige Voraussetzungen an die Umgebung[?]. Da Neo4j eine Java VM muss zb. für Ubuntu das OpenJDK 8 installiert sein. Die Neo4j Instanz wird auf einer Ubuntu Umgebung installiert. Dafür müssen jedoch erst einige Abhängigkeiten und Pakete installiert werden. Um den Aufwand gering zuhalten, wird eine Amazon Web Services EC2 Instanz hochgefahren mit einem speziellen Cloudformation Template, welches alle System Voraussetzungen erfüllt. Cloudformation erleichtert das Erstellen von Umgebungen mit wichtigen Laufzeitparametern. Das folgende Schaubild veranschaulicht alle wichtigen Parameter:

[Schaubild fehlt](#)

### 3 Neo4j

Über den freigegebenen Port 7474 bietet Neo4j eine Weboberfläche zur Abfrage der Daten und Visualisierung der Ergebnisse an. Die Datenbank soll nun initial mit realen Daten befüllt werden.

## 6.3.1 RDF Import

Neo4j lässt sich über Plugins erweitern. Diese so genannten *Prozeduren* werden in Java geschrieben und können mit der Cypher Query Language aufgerufen werden. Mit dem Plugin *neosemantics* können Daten aus einem Triplestore in Neo4j geladen werden. Über die Weboberfläche kann mit einer Cypher Abfrage die Methode zum importieren der Daten aufgerufen werden.

```
1 CALL semantics.importRDF("file:///.../esco_skos.rdf", "RDF", {
    shortenUrls: true, typesToLabels: true, commitSize: 9000,
    languageFilter: 'de'})
```

Listing 2: importRDF Prozeduraufruf

## 6.4 API Gateway

Das API Gateway wird hier als Proxy verwendet, mit dessen Hilfe HTTP Requests an Lambda Funktionen weiter gereicht werden. Die Lambda Funktionen wiederum rufen innerhalb eines virtuellen Netzwerks eine Datenbankabfrage für die EC2 Neo4j Instanz auf. Die URL für diese Instanz ist nur innerhalb des Virtuellen Netzwerks erreichbar, und ist somit für die Anwendung vor dem Proxy verborgen. Innerhalb der Lambda Funktion kann dann mit den berechneten Daten ein Response Objekt erstellt werden und an den Aufrufer der API zurückgesendet werden.

## 6.5 Serverless




Mit den Amazon Web Diensten können serverlose Anwendungen geschrieben werden, die es dem Entwickler ermöglichen sich auf den Kern der Applikation, den Quellcode zu konzentrieren ohne sich Gedanken über Ressourcen und Kapazitäten zu machen. Bereits in der Entwurfsphase konnte fest gestellt werden, dass die Komplexität der Anwendung weniger Programmieraufwand, sondern im finden und implementieren von Algorithmen liegt. Dennoch ist das Bereitstellen von einigen Ressourcen wie einer Laufzeitumgebung und einer API notwendig, um anderen Anwendungen über das Internet das Ausführen des Codes mit Parametern zu ermöglichen. Dabei spielt es keine Rolle wieviele Anwendungen auf gleichzeitig auf die API zugreifen, die Rechenkapazität werden automatisch angepasst. Draus ergibt sich ein weiterer Vorteil für Entwickler und Betreiber. Code wird nicht ausgeführt, wird verschluckt keine Ressourcen, und wird von Ressourcen Dienstleister auch nicht in Rechnung gestellt. Der geringe Overhead, die geringen Betriebskosten und die geringe Komplexität der zu entwickelnden Anwendungen laden daher dazu ein, den serverlosen Ansatz zu wählen. Automatisches Kapazitäts Management Only Pay what you use


Fehlt hier was???

## 6 AWS Lambda

Lambda bietet die Möglichkeit Code auszuführen ohne die nötigen Ressourcen managen zu müssen. Lambda ist Ereignis gesteuert, was bedeutet der Code wird als Reaktion auf ein Ereignis ausgeführt.

### 6.5.2 Serverless Framework

 Das Serverless Framework ist ein Kommandozeilen Tool, und wurde zur Unterstützung des Entwicklungs und Deployment Prozess von serverlosen Anwendungen entwickelt. Absatz Der Quellcode zur Datenverarbeitung in den Lambda Funktionen kann mit diesem Framework nicht nur lokal getestet werden, sondern kann auch direkt mit API Gateway verknüpft werden. Alle wichtigen Parameter wie Umgebungsvariablen zb. Zugangsdaten, URLs und Endpunkte, werden in einer Konfigurationsdatei eingetragen. Über die Kommandozeile kann die Lambda Funktion aufgerufen werden, sowohl lokal als auch auf dem Stage System. Ein Problem  nun an den Tag tritt ist die lokale Nichtverfügbarkeit von Diensten die nur in der Cloud existieren. Nimmt  zb. Änderungen im Code der Lambda Funktion vor und möchte diese testen, muss der Code zunächst neu deployt werden bevor neue Anfragen an die API gesendet werden können. Dieser Prozess braucht Zeit und verlängert den Entwicklungsprozess. Besser wäre es eine lokale simulierte API zur Verfügung zu haben mit welcher sich die Änderungen im Quell Code der Lambda Funktion direkt testen lassen. Dieses Feature wird über Plugins des *Serverless Frameworks* zur Verfügung gestellt.

API implementieren Was ist Api Gateway? Welche Vorteile bietet API Gateway 

## 6.6 Architektur im Überblick

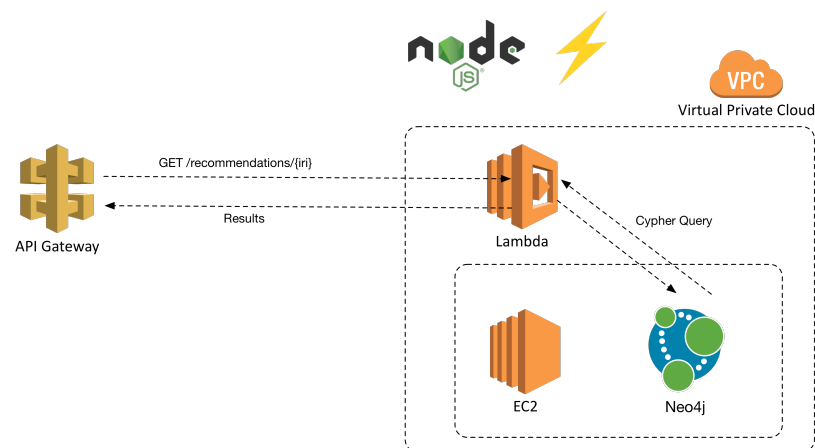



Abbildung 5: Architektur 

Nachdem nun alle Komponenten eingerichtet sind erfolgt das Implementieren der Algorithmen und Datenbankabfragen. Als Laufzeitumgebung für die Lambda Funktio-



nen wird Node.js in der Version 6.10 gewählt. Für die Anbindung an die Neo4j wird die *neo4j-driver* Bibliothek geladen. Damit lässt sich zunächst für den gesamten Scope der Lambda Funktion der Datenbanktreiber der *neo4j* Klasse instanzieren. Wird nun beim ausführen des Codes durch Lambda **wird** die *Handler* Funktion aufgerufen, kann für den jeweiligen Handler eine neue Session geöffnet werden und ein String für die Datenbankabfrage, inklusive Parameter geformt werden. Ist die Abfrage beendet wird die Callback Funktion aufgerufen und ein Response Objekt erstellt, welches das Ergebnis der Datenbankabfrage im JSON String Format enthält.

Die Im Entwurf beschriebene Kern Komponente für **Die** Datenbankanbindung, die **Ein** **uns** **A** **sgabe** von Daten und die Implementierung von **Algorithmen**, soll nun Lambda Funktionen in einer Node.js Laufzeitumgebung implementiert werden. Als erstes wird eine Instanz des Datenbanktreibers erzeugt welche von mehreren Handlern benutzt werden kann. Für jede Anforderung der Engine, soll ein eigener Handler implementiert werde, der genau dann ausgeführt wird wenn ein HTTP Request auf dem zu dem Handler gehörenden Pfad aufgerufen wird. Parameter aus dem Query String, oder Daten die mittels POST übermittelt werden können aus dem *event* Objekt gelesen werden. **Absatz**

Innerhalb des Handlers wird ein *session* Objekt erzeugt, welches **Die** Verbindung zur Datenbank herstellt und über diese auch die Abfragen an die Datenbank erfolgen. Dazu wird die *run* Methode auf dem Objekt aufgerufen und der Querystring und Queryparameter übergeben. Es wird ein Promises erzeugt, auf welchem eine Callbackfunktion aufgerufen wird sobald die Abfrage alle Ergebnisse geliefert hat. Aus dem Resultset kann ein Objekt erzeugt werden welches mit der *callback* Funktion des Lambda Handlers an den Aufrufer, in diesem Fall die API zurück gegeben wird.

Mit dem folgenden Handler wird ein Node *n* über ein eindeutiges Attribut wie einer Id oder eines IRI in der Datenbank abgefragt und an die API zurückgeliefert.

```

1      'use strict';
2
3      var neo4j = require('neo4j-driver').v1;
4
5      var driver = neo4j.driver("bolt://" + process.env.NEO4J_URL, neo4j.auth.
        basic("user", "password"));
6      module.exports.getNode = function(event, context, callback) {
7          var session = driver.session();
8          session
9              .run('MATCH (n:{queryParam}) return n;', {queryParam: event.
                queryStringParameters.label})
10             .then(function (result) {
11                 result.records.forEach(function (record) {
12                     callback(null, {
13                         statusCode: 200,
14                         body: JSON.stringify({

```

```
15         message: record,
16         input: event,
17     }},
18     });
19 });
20 session.close();
21 driver.close();
22 })
23 .catch(function (error) {
24     console.log(error);
25 });
26 };
```

Listing 3: Lambda-Handler Funktion

## 6.7 Deployment mit dem Serverless Framework

Beim Erzeugen eines neuen Projektes mit dem Serverless Framework, wird neben einem leeren Handler auch eine Konfigurationsdatei mit dem Namen *serverless.yml* erzeugt. In dieser werden neben der Route die die Funktion auslöst auch die Art der HTTP Methoden eingetragen. [Absatz](#)

```
1
2 functions:
3   getNode:
4     handler: handler.getNode
5     events:
6       - http:
7         path: node
8         method: get
```

Listing 4: Serverless Konfiguration

Die notwendigen Paketabhängigkeiten für die Datenbankanbindung an Neo4j werden mit *npm install* in den node modules Ordner geladen. Über das Serverless Kommandozeilen Programm, wird mit dem *deploy* Aufruf der gesamte Inhalt des Projektes, inklusive Abhängigkeiten gepackt, Cloudformation Templates erstellt und dann auf ein S3 Bucket in der AWS Cloud geladen.

## 6.8 Algorithmen und Metriken

Das technische Gerüst ist nun komplett. Kompetenzen können per GET Request an die API, und unter Angabe ihrer IRIs abgefragt werden. Jedoch sind die Anforderungen an das Empfehlungssystem mittels Metriken und Algorithmen Ähnlichkeiten zu bestimmen.

## 1 Shortest Path

Neo4j bringt eine Implementierung von kürzesten Pfaden direkt von Haus aus mit. Dafür müssen in der *MATCH* Klausel nur ein spezifischer Anfangsknoten *s* und ein Endknoten *t* abgefragt werden und diese beiden in der *shortestPath* Funktion als Parameter übergeben. Wahlweise kann der Pfad oder die Länge des Pfades zurückgegeben werden. [Absatz](#)

Als grobe Metrik kann die *shortestPath* Funktion dazu verwendet werden um heraus zu finden wie weit eine Kompetenz von einer anderen entfernt ist. Je näher, desto ähnlicher.

## 2 Jaccard Index

Zunächst soll geprüft werden ob es Knoten Typen und Beziehungen im Graphen gibt mit denen sich eine Schnittmenge für zwei Kompetenzen berechnen lässt. Dabei soll eine Kompetenz von außen über die API als Eingabe erfolgen und solche Kompetenzen die eine Schnittmenge mit der gegebenen haben in geordneter Reihenfolge zurückgegeben werden. Diese Metrik lässt sich mit einer Cypher Abfrage direkt implementieren ohne weiteren Code ausführen zu müssen. Dazu wird zunächst die Abfrage formuliert.

*Welche Kompetenzen haben wieviele gemeinsame Beziehungen vom selben Typ zur gegebenen Kompetenz* [Absatz](#)

Als erstes werden mit einer **MATCH** Klausel alle Knoten *c<sub>gesucht</sub>* gesucht, die neben dem gegebenen Knoten *c<sub>input</sub>* eine eingehende Beziehung zu einem Knoten *o* vom Typ *ns4\_Occupation* haben und die Schnittmenge bilden. [Absatz](#)

```
1 MATCH (c:ns4_Skill {ns5_prefLabel:{queryParam}})-[:ns5_related]->(o:
    ns4_Occupation)<-[:ns5_related]-(other:ns4_Skill)
```

Die Kompetenz aus dem Input wird mittels Parameter der Query übergeben und das Resultat des Patterns in den Variablen *c*, *o*, *other* mit der **WITH** Klausel in den nächsten Query Teil weiter gereicht. Mit der Aggregatsfunktion **COUNT** wird dann der Betrag der Schnittmenge von Knoten vom Typ *ns4\_Occupation*, die eine eingehende Beziehung vom Typ *ns5\_related* von Knoten *c<sub>input</sub>* und Knoten *c<sub>gesucht</sub>* vom Typ *ns4\_Skill* haben. [Absatz](#)

```
1 MATCH (c:ns4_Skill {ns5_prefLabel:{queryParam}})-[:ns5_related]->(o:
    ns4_Occupation)<-[:ns5_related]-(other:ns4_Skill)
2
3 WITH c, other, COUNT(o) AS intersection, COLLECT(o.ns5_prefLabel) AS i
```

Als nächstes müssen die beiden Mengen in separaten Variablen *s<sub>1A</sub>* und *s<sub>2B</sub>* gespeichert werden,

```

1 MATCH (c:ns4_Skill {ns5_prefLabel:{queryParam}})-[:ns5_related]->(o:
   ns4_Occupation)<-[:ns5_related]-(other:ns4_Skill)
2
3 WITH c, other, COUNT(o) AS intersection, COLLECT(o.ns5_prefLabel) AS i
4 MATCH (c)-[:ns5_related]->(c_other:ns4_Occupation)
5
6 WITH c,other,intersection, i ,COLLECT(c_other.ns5_prefLabel) AS s1
7 MATCH (other)-[:ns5_related]->(oc:ns4_Occupation)
8
9 WITH c,other,intersection,i, s1, COLLECT(oc.ns5_prefLabel) AS s2

```

um dann mittels Listenfunktion **FILTER** die Vereinigungsmenge zu berechnen. [Absatz](#)

```

1 MATCH (c:ns4_Skill {ns5_prefLabel:{queryParam}})-[:ns5_related]->(o:
   ns4_Occupation)<-[:ns5_related]-(other:ns4_Skill)
2 WITH c, other, COUNT(o) AS intersection, COLLECT(o.ns5_prefLabel) AS i
3
4 MATCH (c)-[:ns5_related]->(c_other:ns4_Occupation)
5 WITH c,other,intersection, i ,COLLECT(c_other.ns5_prefLabel) AS s1
6
7 MATCH (other)-[:ns5_related]->(oc:ns4_Occupation)
8 WITH c,other,intersection,i, s1, COLLECT(oc.ns5_prefLabel) AS s2
9
10 WITH c,other,intersection,s1+filter(x IN s2 WHERE NOT x IN s1) AS
   union, s1, s2
11
12 RETURN c.ns5_prefLabel, other.ns5_prefLabel, s1,s2,intersection,((1.0*
   intersection)/SIZE(union))
13
14 AS jaccard ORDER BY jaccard DESC LIMIT 10

```

Der Quotient aus dem Betrag der Schnittmenge, und dem Betrag der Vereinigungsmenge ergibt dann den Jaccard Index und wird gemeinsam mit den Namens Attributen für die die Kompetenzen  $c_{input}$  und  $c_{gesucht}$  in, in sortierter Reihenfolge in der **RETURN** Klausel in das Ergebnis geschrieben.

### 6.8.3 Levensthein oder Edit-Distanz

[11pt,fleqn]article **Ende**