Bachelorarbeit am Institut für Informatik der Freien Universität Berlin

Human-Centered Computing (HCC)

Konzeptannotation von Ideen basierend auf WordNet

Ingrid Gancia Tchilibou Boudjeka

Matrikelnummer: 5020686

ganciatchilibou@yahoo.fr

Betreuer: Herr Maximilian Mackeprang

Erstgutachterin: Prof. Dr. C. Müller-Birn

Zweitgutachter: <Name des Zweitgutachters>

Berlin, 12. Januar 2020

b

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, den 12. Januar 2020

Ingrid Tchilibou

Zusammenfassung

In dieser Arbeit werden <Hier sollten Sie eine kurze, aussagekräftige Zusammenfassung (ca. eine Seite) Ihrer Arbeit geben, welche das Thema der Arbeit, die wichtigsten Inhalte, die Arbeitsergebnisse und die Bewertung der Ergebnisse umfasst.>

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Vorwort

Allgemeine Hinweise zur Erstellung einer Abschlussarbeit

- Beachten Sie, dass diese Vorlage für einen zweiseitigen Ausdruck angelegt wurde.
- Über die Papierqualität können Sie entscheiden, aber wir empfehlen aber Seiten mit wichtigen, farbigen Grafiken auch in Farbe auszudrucken und dabei ein höherwertiges Papier zu verwenden.
- Bitte stimmen Sie mit dem Betreuer Ihrer Arbeit auch den Zweitgutachter ab. Die Anfrage des Zweitgutachters erfolgt von Ihnen. Es ist an dieser Stelle sinnvoll, die Anfrage mit einer kurzen Zusammenfassung der Arbeit zu stellen.
- Bitte beachten Sie, dass Sie Ihre Abschlussarbeit mit einer Klebebindung versehen, eine Ringbindung ist nicht erwünscht.

1 Einleitung

1.1 Thema und Kontext

Ideen im großen Stil (Engl.:Large Scale Ideation) beschreibt die Anwendung von semantischen Technologien in einem Mensch-Computer-Kontext, mit dem Ziel innovative und einzigartige Ideen zu generieren und zu fördern. Allerdings bringt sie seine eigenen Probleme mit sich, allem voran der Umgang mit großen Mengen an Ideen. Diese macht es für Forscher sehr schwer, die Ergebnisse einer Herausforderung (Sammlung von Ideen zu einem bestimmten Thema) auszuwerten, insbesondere wenn sie manuell erfolgt. Ein Beispiel ist das "Ideas-to-Market"-Projekt[Com20], in dem Ideen zu neuen Technologien gesammelt werden. Im Anwendungsfall "Bionic Radar" mussten die Organisatoren insgesamt 581 Ideen lesen um diese in einen strukturierten Raum übergeordneter Ideen (Hypernyme) einzubetten/um sich einen Überblick über den entstandenen "Ideen-Raum" zu verschaffen. Es gibt in der Informatik verschiedene Ansätze, um Ideen für Computer analysierbar zu machen, welche die Verarbeitung dieser Daten in Zukunft stärker automatisieren könnten. Ein Beispiel ist die Verwendung von sogenannten Satz-Einbettungen (Sentence-Embeddings), die Ideen-Texte in einen hochdimensionalen Raum einordnen. Ein Problem an diesem Ansatz ist allerdings, dass Informationen über die Strukturen der Ideen nicht mehr abrufbar sind, beispielsweise der Abstraktionsgrad einer Idee. Ein weiteres Problem ist die Word-Sense-Disambiguierung: Eine Nennung des Begriffs "Keyboard" könnte entweder ein Musik-Instrument oder ein Computer-Eingabegerät meinen. Satz-Einbettungen können diese semantische Uneindeutigkeit nicht auflösen.
Das sogenannte Interactive Concept Validation (ICV)[MMBS19] bietet einen Ansatz, um Ideen mit semantisch eindeutigen Konzepten anzureichern. Ein Beispiel für die praktische Umsetzung dieses Ansatzes ist z.B. das vom Human Computing Center (HCC) der Freien Universität Berlin entwickelte ICV-Tool. Dabei wurden für das Tool bisher Dbpedia als Datenquelle für ICV und Wikidata für die Superklassen verwendet. Wikidata-Superklassen haben aber den Nachteil, dass sie von Freiwilligen erstellt werden. Dadurch ist die Qualität der Superklassen sehr unterschiedlich. Außerdem können Superklassen in Wikidata Zyklen bilden, was die algorithmische Analyse schwer macht. Das an der Princeton University entwickelte Wordnet hat dieses Problem nicht, da es von Experten erstellt wird. Dadurch sind die Beziehungen zwischen Konzepten eindeutiger definiert (und zum Beispiel zyklenfrei).

1.2 Zielsetzung der Arbeit

Ein wichtiger Schritt im Prozess von Ideas "Ideas-to-Market" ist es, gesammelte Ideen zunächst zu ordnen und zu kategorisieren. Hierbei könnten ICV-Methoden, welche die Stärken von Wordnet nutzen, sinnvoll sein. Die Hauptforschungsfrage für diese Arbeit war daher: Können Ideen mit Hilfe von Wordnet und Hypernymen kategorisiert werden? Bzw. Wie können Ideen mit Hilfe von Wordnet annotiert und mit Hilfe von Hypernymen kategorisiert werden?
Diese Frage soll beantwortet werden, indem ein Beispieldatensatz manuell annotiert und dann analysiert wird. Das Hauptziel wird in drei untergeordnete Fragestellungen unterteilt:

1. Wie kann WordNet in die bestehende Software integriert werden?
2. Wie kann das Hypernym eines Wortes in WordNet bestimmt und extrahiert werden?
3. Wie können aus WordNet Hypernym-Kategorien für Ideen erstellt werden?

1.3 Vorgehen bei der Umsetzung

Die Eingabe soll aus Ideen aus dem oben erwähnten Anwendungsfall "Bionic Radar" bestehen, um einen Vergleich zwischen manueller und automatisierter Kategorisierung zu ermöglichen. Der Arbeitsprozess gliedert sich dabei in fünf Hauptphasen mit entsprechenden Unterschritten:

1. Um Ideen-Texte mit Wordnet zu verknüpfen, wird ein neues Backend für das "Interactive Concept Validation" ICV Tool des HCC implementiert. Dieses Backend weist folgende Funktionalitäten auf:
   - Tokenization: Wörter, Sätze und Absätze werden erkannt und gruppiert.
   - Stop Word Filtering: Wörter, die nicht dazu beitragen den Text zu verstehen, werden eliminiert.
   - Bedeutungs-Kandidaten (Synset-Candidates) werden gesucht, indem die Synset-ID und die Beschreibung des Synsets aus WordNet übernommen wird.
2. Mit der neuen Implementierung werden die Synsets von Ideen gesammelt.
3. Für die annotierten Ideen werden die Hypernyme aus Wordnet extrahiert.
4. Das Ergebnis der Annotation und Extraktion werden visualisiert, um zwei Aspekte zu untersuchen:
   - Wie viele Top-Level Hypernyme (direkt unter "Entity") kommen im Datensatz vor?

- Wie viele 2nd-Level Hypernyme kommen vor, und pro Hypernym, wie viele Ideen werden damit beschrieben?

Für die Visualisierung gibt es zwei Möglichkeiten:

- Visualisierung mit Dendogramm: die Beziehungen zwischen Hypernymen werden als Hierarchie angezeigt.[1]
- Visualisierung durch ein Balkendiagramm: Die Anzahl der Ideen pro Hypernym werden gezählt.

Die annotierten und mit Hypernymen versehenen Ideen können außerdem die Grundlage für weitere, zukünftige Visualisierungen bilden.

1.4 Aufbau der Arbeit

Diese Arbeit besteht aus sechs Hauptkapiteln. Nach einer allgemeinen Einführung werden in Kapitel 2 eine kurze Erläuterung der schon existierenden Methoden und Abgrenzungen, sowie eine Einführung in WordNet vorgenommen. Danach wird in Kapitel 3 der Entwurf von allen Elementen, die für die Realisierung dieses Projekts notwendig sind, sowie die zu verwendenden Werkzeuge detailliert beschrieben. Zusätzlich wird im gleichen Kapitel die API dokumentiert, die die Annotation der in den Ideen enthaltenen Wörter erlaubt, sowie die Code-Struktur mit einigen wichtigen Teilen des Codes, die die Annotation eines Textes mit WordNet erlaubt. Des Weiteren wird zuerst in Kapitel **??** Anwendungsfall, der Anwendungsfall Bionic Radar beschrieben und danach mit Hilfe von der ICV und der erstellte WordNet-API in der Kapitel 3 annotiert. Darüber hinaus werden wir in Kapitel **??** die in Kapitel **??** erhaltenen Elemente mit Hilfe des Hypernyms kategorisieren und anschließend mit Hilfe eines Dendrogramms und eines Balkendiagramms visualisieren. Um schließlich festzustellen, ob unser Ziel erreicht wurde, werden wir in Kapitel 6 einen Test an einigen Personen durchführen und einen kleinen Vergleich des erhaltenen Ergebnisses mit dem bereits vorhandenen zusammen mit dem Ergebnis der manuellen Kategorisierung durch die HCC-Gruppe durchführen. Wir schließen mit einer kurzen Schlussfolgerung. Eine graphische Übersicht über einen Aufbau dieser Arbeit ist in der Abbildung 1.1 dargestellt.
Bei den nachfolgend verwendeten, persönlichen Ausdrücken ist die männliche Schreibweise gewählt worden, damit die Lesbarkeit der Arbeit erhöht wird. Darüber hinaus werden eine Reihe englischer Begriffe verwendet, um einerseits dem interessierten Leser das Studium der häufig vorhandenen englischen Original-Literatur zu erleichtern und andererseits eine Verzerrung bestehender Fachbegriffe durch die Übersetzung zu vermeiden. Vom herkömmlichen Text werden diese Begriffe durch Kursivschrift unterschieden.

[1] `https://de.wikipedia.org/wiki/Hierarchische_Clusteranalyse`

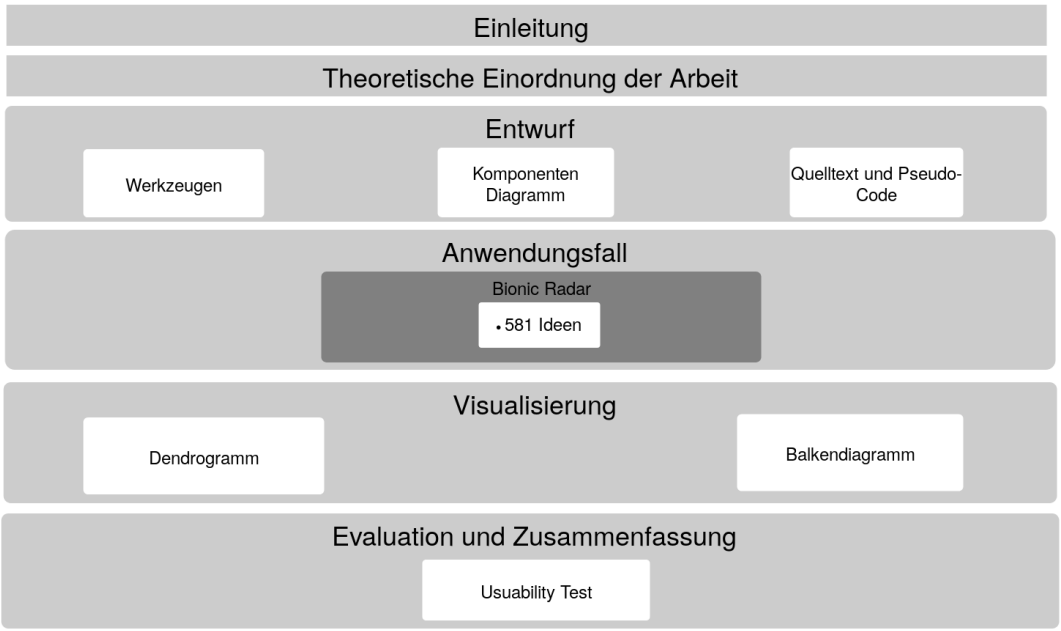


Abbildung 1.1: Aufbau der Arbeit (vgl. Beschreibung Abschnitt 1).

2 Theoretische Einordnung der Arbeit

In diesem Kapitel wird sowohl die alte Methode zur Lösung unseres Problems als auch eine kurze Einführung in die neue Wordnet-Methode gegeben.

2.1 Bisherigen Methode und Abgrenzung

Durch die große Anzahl an generierten Ideen in Large Scale Ideation Challenges ist es ökonomisch nicht sinnvoll, alle Ideen im Detail zu lesen.
Laut Siangliulue[Sia15] ist ein weiteres Problem, dass Teilnehmer an Challenges oft profane und repetitive Ideen einreichen.
Ein Ansatz dieses Problem abzumildern ist es, Ideen schon während der Generierung zu kategorisieren. Beispielsweise wird im CrowdMuse System von Girotto et al.[GWB19] die manuelle Kategorisierung genutzt, um eine Matrix-Visualisierung des Ideen-Raumes zu erzeugen. Die Manuelle Kategorisierung in diesem Projekt passierte aber auf reiner Wort-Basis, ohne zusätzliche Kontextinformationen (wie z.b. Hypernyme oder Superklassen).
Bei der semantischen Anreicherung von Ideen handelt es sich um einen Ansatz, Ideen für Computer verständlich zu machen, indem Terme (einzelne Wörter wie "Keyboard" oder zusammengesetzte Wörter wie "pet food") in den Ideentexten mit eindeutigen Konzepten in einem Knowledge-Graph (KG) verbunden

werden. Ein Knowledge Graph organisiert verschiedene reale Einheiten, sogenannte Konzepte, mit ihren Beziehungen in einem Graphen. Es stellt auch ein Schema zur Verfügung, das diese Konzepte in Klassen (abstrakte Konzepte) gruppiert, die auch Beziehungen zueinander haben[RP17]. Dieses beschreibt die Ähnlichkeiten zwischen Konzepten durch einen reicheren Satz von Beziehungen. (Bsp. : 'is-hypernym,' 'is-holonym,' 'is-a-part-of,' 'is-synonym') [MMBS19] [RP17]
Ein Beispiel für die Erfolgreiche Anreicherung von Ideen im Kontext von Inspiration von Designern findet man bei Gilon et al.[GCN+18]

2.2 WordNet 3.1 : "A Lexical Database for Englisch"

Um das Problem der manuellen Kategorisierung zu lösen, werden wir in dieser Arbeit die WordNet-Datenbank verwenden, um Ideen automatisch zu annotieren und zu kategorisieren. WordNet[Mil98] ist ein freies, kostenloses, semantisches Netzwerk, das aus einer umfangreichen lexikalischen Datenbank in englischer Sprache besteht. In WordNet gibt es mehrere Arten von Konzepten: Begriffe (aus dem Lexikon) und die Bedeutung der Begriffe (genannt Synset). Die Zusammenfassung von Namen, Verben, Adjektiven und Adverbien in eine Gruppe von Kognitiven Synonymen (Synsets), um ein bestimmtes Konzept auszudrücken, sind sehr gut geeignet, um die semantischen und lexikalischen Beziehungen von Wörtern miteinander zu verbinden[Uni09]. Diese ermöglicht es, ein Wort in einem Kontext zu Klassifizieren.
Zum Beispiel kann das Wort "Framework" entweder eine hypothetische Beschreibung einer komplexen Einheit oder eines komplexen Prozesses[Synset 01], die zugrunde liegende Struktur[Synset 02] oder eine Struktur, die etwas trägt oder enthält[Synset 03], sein. Anstatt Konzepte um ihre lexikalische Form herum zu gruppieren, gruppiert WordNet durch Synsets Konzepte nach ihrer Bedeutung im Kontext. WordNet hat dadurch zwei Arten von Beziehungen:

- Zwischen einem Synset und den Begriffen, mit denen es im Zusammenhang bezeichnet wird.
- Zwischen einem Synset und seiner Definition im Kontext.

Darüber hinaus gibt es eine weitere Art der Beziehung zwischen Synsets:

- Hypernymie (Hyperonymie) und Hyponymie: Hypernym ist wie der Oberbegriff eines Wortes X. Wenn Y in X enthalten ist, dann ist Y das Hyponym von X und X ist das Hypernym von Y. Z.B. sind 'Hund', 'Katze', 'Schaf', 'Maus' Hyponyme von 'Tier' und 'Tier' ist Hypernym von 'Hund'.
- Meronymie und Holonymie: Meronym ist ein Wort X dessen Bedeutung einen Teil eines Wortes Y bezeichnet. Holonyme sind das Gegenteil davon. Z.B. ist 'Blatt' ein Meronym von 'Baum' und 'Baum' ist ein Holonym von 'Blatt'.

Für das Annotieren von Synsets, die in Ideen-Texten vorgefundenen wurden, liegen prinzipiell zwei Möglichkeiten der Analyse nahe. Zum einen kann für eine konkrete Idee ein "Abstraktionsgrad" errechnet werden (indem die Anzahl an Hypernymen für eine Idee berechnet wird), der es ermöglicht, die Eignung der Idee als Inspiration zu bewerten[CDD16]. Zum anderen könnten die Hypernyme eine Möglichkeit geben, Ideen zu kategorisieren, indem jede Idee einem abstrakten Hypernym zugeordnet wird. Diese Kategorisierung könnte helfen, Teilnehmer*innen zu orientieren, oder Ideen-Räume automatisiert zu unterteilen. Deswegen benutzen wir für die Kategorisierung die Hypernyme aus WordNet. Auf Wordnet werden bis auf ein paar Verbsynsets alle Synsets durch das Synset "entity" subsumiert.Dies wird als die höchste Abstraktionsebene angesehen. In diesem Paper[GG18] untersucht der Autor die Semantik von Wörtern auch mit Wordnet und verwendet die "level of abstractionäls Metrik. Der Autor beschreibt diese auf Deutsche übersetzt als " Die Abstraktionsebene steht in der Taxonomie in einem negativen Zusammenhang mit der Tiefe des Substantivs, und zwar so, dass das Stammsubstantiv Ëntität"das abstrakteste ist, während die tiefsten Substantive in der Taxonomie am wenigsten abstrakt sind [30]. Das Komplement der Abstraktionsebene zur Einheit ist ein Maß für die Konkretheit des Wortes." Die Abbildung[2.1] zeigt exemplarisch die Hypernym-Beziehung zwischen zwei Wörtern "Door" und "Windows". Laut dieser Abbildung werden diese Wörter in "*Structur Constructor*" kategorisiert.

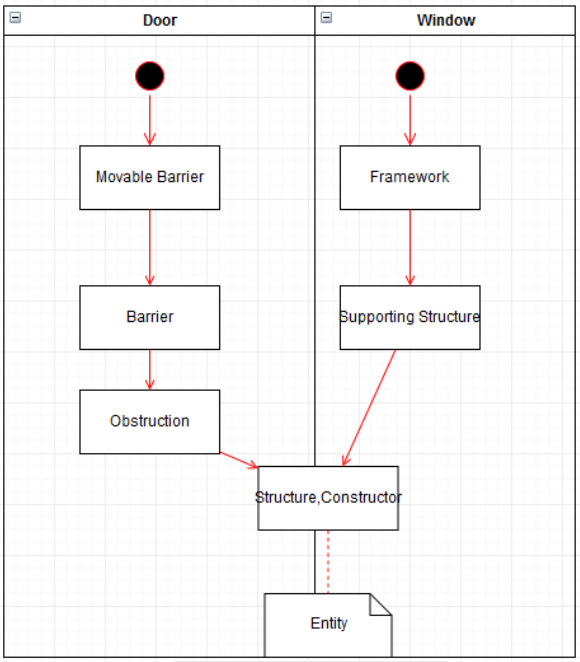


Abbildung 2.1: Hypernym 2

3 Entwurf

In diesem Kapitel werden wir zuerst die Elemente besprechen, die für die Realisierung dieses Projektes verwendet wurden, und dann die Annotation der Ideen mit Hilfe der Wordnet-API detailliert beschreiben. Das folgende Komponentendiagramm beschreibt die komplette Vorgehensweise, um unser Ziel zu erreichen.

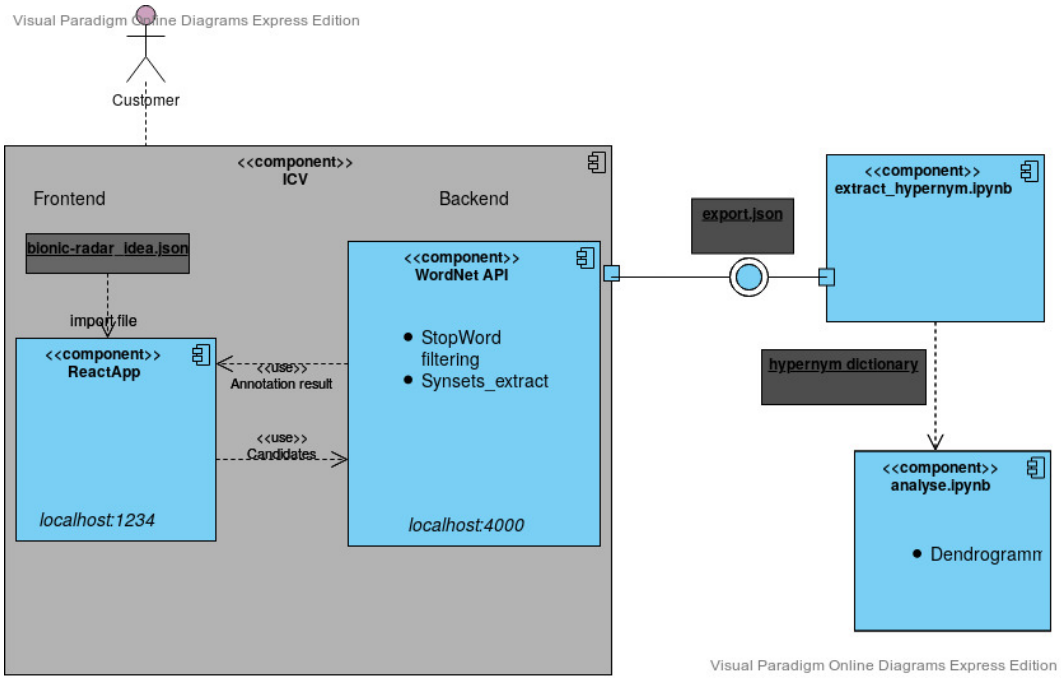


Abbildung 3.1: Komponenten Diagramm

3.1 Werkzeuge

Für die Realisierung der WordNet API, die in das ICV Backend integriert wird, werden wir als Programmiersprache Python verwenden, genauer gesagt die Version 3.6.8 Aufgrund der zahlreicher Bibliotheken für Sprachverarbeitung wie z.B. NLTK ist Python eine geeignete Programmiersprache für die Mensch-Maschine-Interaktion und das Textstudium. Da Python die Sprache ist, die vom Autor und Entwickler der Arbeit am Besten beherrscht wird, haben wir uns entschieden, Python für die Realisierung dieses Projekts zu verwenden. Daneben wird jupyter notebook als Editor eingesetzt und dient auch zur Vi-

sualisierung der Ergebnisse. Für die Sicherung und Verwaltung des Projekts wird der Webserver für Software-Projekte Github benutzt.

3.2 WordNet-API: NLTK-Bibliothek

Eine API (Application Programming Interface) ist eine Programmierschnittstelle, die den Informationsaustausch zwischen einer Anwendung und einzelnen Programmteilen ermöglicht. Die von uns entwickelte Schnittstelle ermöglicht es anderer Software, durch Anfragen transparent auf die Wordnet-Daten zuzugreifen, ohne diese zu verändern. Die Antwort der Anfragen ist eine JSON-Datei. JSON (JavaScript Object Notation) ist ein kompaktes Datenformat, das Informationen in einer für den Menschen lesbaren und verständlichen Form enthält. In unserem Fall, ruft das ICV-Backend die WordNet-API auf, um Annotationen von Ideen zu erhalten. Für die Realisierung dieser WorNet-API haben wir das Mini-Web-Framework flask von python gewählt, das durch seine vielfältigen Elemente die Entwicklung der Anwendungen erleichtert [AMM+15]. Die Kommunikation zwischen der WordNet API und dem ICV-Backend wird über CORS abgewickelt. CORS ist in der Bibliothek flask_cors enthalten, die die Kommunikation zwischen beiden Anwendungen ermöglicht. CORS und FLASK werden wie unten angezeigt im Programm aufgerufen:

```
1 from flask import Flask,
2 from flask_cors import CORS
3 app = Flask(__name__)
4 CORS(app)
```

Listing 3.1: Anwendung von CORS und FLASK

Die Annotation der Ideen erfolgt dann über die in der NLTK-Bibliothek enthaltene Wordnet-Bibliothek. NLTK (Natural Language Toolkit) ist eine Suite von Open-Source-Programmmodulen, Tutorials und Problemsets, die gebrauchsfertige Computerlinguistik-Kursmaterialien zur Verfügung stellen [LB02].
Die WordNet-API läuft im *localhost* unter Port 4000, empfängt einen Text (eine Idee) und gibt bei der Ausgabe eine Json-Datei zurück. Diese Json-Datei hat folgende Struktur:

```
{
   "annotation_candidates":[
      {
         "offset": "Offset des Wortes in Idee Text",
         "resource_candidates":[
            {
               "description": "Beschreibung des Synset",
               "label": "Synset Name",
               "offset": "Offset des Wortes in Idee Text",
               "resource": "Id des Synset Name in WordNet",
               "source": "Synset name in WordNet",
               "text" : "Ursprunchliches Wort",
               "pos": "Wortart(part of speech)"
            },
            {...},
            {...},
         ],
         "text": "Ursprunchliches Wort"
      }
      {...}
      {...}
   ],
   "text": "Idee Text"
}
```

Abbildung 3.2: JSON Ausgabestruktur der WordNet-API

Das in Abbildung 3.2 sichtbare Feld *annotation candidates* [3.2] sind alle Wörter, die in der in den Parametern angegebenen Idee enthalten sind. Stopp-Wörter wie ändortheünd Satzzeichen sind nicht enthalten. Diese Wörter sowie Symbole, die keine nützlichen Informationen zum Verständnis der Idee enthalten, wurden wie unten detailliert mit Hilfe der NLTK Stopword-Bibliothek gefiltert.

```
1  import nltk
2  from nltk.corpus import stopwords
3  import re
4
5  def stop_words_filtering(text):
6      reg_exp = r"[a-zA-Z]+" #Regulaeare Ausdruck,der alle erlaubten Sylabeln
           enthaelt. Von "a" bis "z" und von "A" bis "Z"
7      stop_words = set(stopwords.words('english')) #stopwoerter fuer englisches Text
8      word_tokens = [ ]
9      a = re.compile(reg_exp)
10     word_tokens = word_tokens + a.findall(text) #loescht die Satzzeichnen im Text
11     filtered_sentence = [w for w in word_tokens if not w in stop_words] #loescht die
           Stopp-Woerter im Text
12     return filtered_sentence
```

Listing 3.2: Unnötigen Wörter im Text Eliminieren

Darüber hinaus stellt ein *ressource-candidate* ein Synset-Element eines *annotation candidate* dar. Daher ist die Länge der *ressource-candidate* proportional zur Anzahl der Synset-Elemente, die in einem *annotation candidate* enthalten sind. Die folgende Funktion ermöglicht es, die Synsets eines Wortes von WordNet mit python zu erhalten:

```
1  from nltk.corpus import wordnet
2  def get_synsets(word):
3      synset_list = wordnet.synsets(word)
4      return synset_list
```

Listing 3.3: Synsets von WordNet mit Python extrahieren

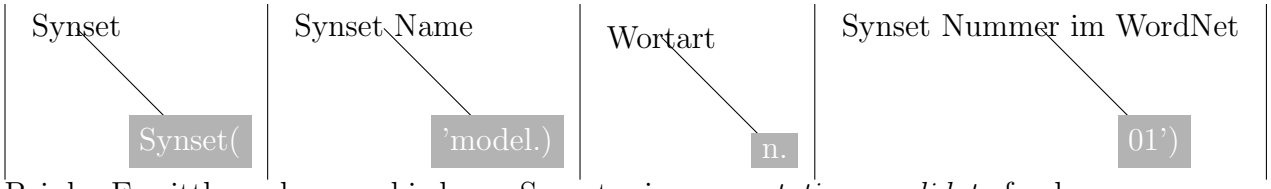
Nehmen wir zum Beispiel wieder das Wort "Framework" und extrahieren wir die Synsets über unsere API:

```
1  get_synsets('Framework')
```

Listing 3.4: Beispiel

Die Ausgabe ist eine Liste von Synset-Instanzen:
[Synset('model.n.01'), Synset('framework.n.02'), Synset('framework.n.03')]

Ein Element(Synset) dieser Liste besteht aus:



Bei der Ermittlung der verschiedenen Synsets eines *annotation candidate* fand die für jedes Synset extrahierte API die folgenden Informationen zum Verständnis des Synsets nützlich:

- Die Beschreibung des Synsets, die wie folgt im Wordnet extrahiert wird:

```
1 def get_definitions(syns):
2     return syns.definition()
```

Listing 3.5: Beschreibung eines Synsets

- Der Name des Synsets (label). Ein Synset hat mindesten ein Label im Wordnet. Wir haben uns entschieden immer das erste Label zu nehmen. Um sicherzugehen das wir immer ein Label haben.

```
1 def get_label(syns):
2     lemmas = syns.lemmas()
3     name = lemmas[0].name()
4     return name
```

Listing 3.6: Label

- die Position des *annotation candidate* derzeit in der Idee in Parameter aufgenommenen annotiert. Dadurch kann das ICV-Tool dem Benutzer jedes Mal mitteilen, welches Wort gerade annotiert wird. Falls sich ein Wort in einer Idee wiederholt, dann ist es nicht offensichtlich, welche dieser Wiederholungen gerade annotiert wird. Aus diesem Grund speichert das Api jedes Wort mit der Anzahl seiner Vorkommen in einem Wörterbuch und prüft bei jeder Behandlung eines Wortes, um welches Vorkommen es sich handelt und bestimmt so seine Position in der Idee mit Hilfe der folgenden Funktion [1].

```
1 def find_nth(text, wort, occurence):
2     start = text.find(wort)
3     while start >= 0 and occurence > 1:
4         start = text.find(wort, start+len(wort))
```

[1] https://stackoverflow.com/questions/1883980/find-the-nth-occurrence-of-substring-in-a-stri

```
5            occurence -= 1
6        return start
```

Listing 3.7: Position eines sich wiederholenden Wortes in einem Text

- Die aktuelle Position id des Synsets in der Datenbank Wordnet("resource"). Dies kann helfen, das Synset direkt in der Wordnet-Datenbank zu finden, ohne das Wort, das es abdeckt, durchzugehen.
- Das Synset selbst ("source"")
- Die *annotation candidate*, die gerade bearbeitet wird ("text").
- Die Wortart("pos"). Die verschiedenen Wortart, die auf Wordnet existieren, sind: Name(n), Verb(v),Adjektiv(a), Adverb(r), Adjektives Satellite(s).

```
1 def get_class(syns):
2     return syns.pos()
```

Listing 3.8: Wortart

Sobald die API alle wichtigen Informationen für das Annotieren eines Wortes mit Wordnet gesammelt hat. Diese Informationen werden in einem Wörterbuch gespeichert. Ein Python-Wörterbuch ist eine Datentyp, der ein elektronisches Wörterbuch darstellt, das mehrere Elemente enthält. Jedes Element des Wörterbuchs besteht aus einem Key und seinem Wert. Es gibt verschiedene Arten von Python-Wörterbüchern. Wir werden für den Aufbau unserer Json-Datei ein *Nested-dictionary* verwenden[Gee20]. Ein *Nested-dictionary* ist eine Zusammenstellung von mehreren Wörterbüchern. Jedes Element des Wörterbuchs besteht aus einem Key, der wiederum ein innere Wörterbuch ist, und einem Wert, der die Elemente des Wörterbuchs repräsentiert und ist wieder ein Wörterbuch.

In unserem Fall haben wir zwei Hauptwörterbücher verdichtet. Ein internes Wörterbuch, das als Key eines *ressource-candidate* und als Wert eine Wörterbuchliste hat, die alle oben aufgelisteten Elemente eines *ressource-candidate* enthält. Und ein externes Wörterbuch, das das interne Wörterbuch umfasst. Der Key ist die *annotation candidates* und der Wert ist eine Liste von Wörterbüchern, wobei jedes einzelne die *annotation candidate* eines Wortes repräsentiert, das in der gegebenen Idee enthalten ist. Ein *Nested-dictionary* ist keine Json-Datei, deshalb ist es notwendig, es in eine Json-Datei zu ändern, damit ein anderes Programm (in unserem Fall das ICV-Tool) durch eine Anfrage an unsere API alle von der API bereitgestellten Informationen erhalten kann.

Und um sie unabhängig von der verwendeten Programmiersprache nutzen zu können. Um ein Wörterbuch in eine Json-Datei zu ändern, verwenden wir die Python-Json-Bibliothek, die die dumps()-Funktion enthält. Dies ermöglicht die Transformation. Diese Funktion wird wie folgt auf unserem fertigen erhaltenen *Nested-dictionary* aufgerufen:

```
1 json.dumps(annotation)
```

Listing 3.9: Wörterbuch im Json-Datei umwandeln

Die Ergebnis davon ist die oben erwähnte Json-Datei [3.2]. Abschließend müssen Sie, um eine Idee mit unserer Wordnet-Api zu annotieren, zuerst unseren Server starten und dann eine Anfrage die wie folgt an aussieht senden:

4 Anwendungsfall: Bionic Radar

5 Visualisierung

5.1 Balkendiagramm

das

5.2 Dendrogramm

6 Zusammenfassung und Ausblick

- Die Zusammenfassung sollte das Ziel der Arbeit und die zentralen Ergebnisse beschreiben. Des Weiteren sollten auch bestehende Probleme bei der Arbeit aufgezählt werden und Vorschläge herausgearbeitet werden, die helfen, diese Probleme zukünftig zu umgehen. Mögliche Erweiterungen für die umgesetzte Anwendung sollten hier auch beschrieben werden.

Literaturverzeichnis

[AMM+15] Fankar Armash Aslam, Hawa Nabeel Mohammed, Jummal Musab Mohd, Murade Aaraf Gulamgaus, and PS Lok. Efficient way of web development using python and flask. *International Journal of Advanced Research in Computer Science*, 6(2), 2015.

[CDD16] Joel Chan, Steven Dang, and Steven P Dow. Comparing different sensemaking approaches for large-scale ideation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2717–2728. ACM, 2016.

[Com20] Human-Centered Computing(HCC). Innovonto/Ideas-to-Market. `https://www.mi.fu-berlin.de/en/inf/groups/hcc/research/projects/innovonto/index.html`, 2020. [Online; accessed 12-Januar-2020].

[GCN+18] Karni Gilon, Joel Chan, Felicia Y Ng, Hila Liifshitz-Assaf, Aniket Kittur, and Dafna Shahaf. Analogy mining for specific design needs. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 121. ACM, 2018.

[Gee20] GeeksforGeeks. Nested-dictionary. `https://www.geeksforgeeks.org/python-nested-dictionary/`, 2020. [Online; accessed 12-Januar-2020].

[GG18] Georgi V Georgiev and Danko D Georgiev. Enhancing user creativity: Semantic measures for idea generation. *Knowledge-Based Systems*, 151:1–15, 2018.

[GWB19] Victor Girotto, Erin Walker, and Winslow Burleson. Crowdmuse: Supporting crowd idea generation through user modeling and adaptation. In *Proceedings of the 2019 on Creativity and Cognition*, pages 95–106. ACM, 2019.

[LB02] Edward Loper and Steven Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[Mil98] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[MMBS19] Maximilian Mackeprang, Claudia Müller-Birn, and Maximilian Timo Stauss. Discovering the sweet spot of human-computer configurations: A case study in information extraction. *arXiv preprint arXiv:1909.07065*, 2019.

[RP17] Daniel Ringler and Heiko Paulheim. One knowledge graph to rule them all? analyzing the differences between dbpedia, yago, wikidata & co. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 366–372. Springer, 2017.

[Sia15] Pao Siangliulue. Supporting collaborative innovation at scale. In *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 9–12. ACM, 2015.

[Uni09] Princeton University. Wordnet: a lexical database for the english language, 2009.

Appendix

6.1 Erster Teil Appendix

6.2 Zweiter Teil Appendix