Fachhochschule Nordwestschweiz

Job Annotations

Bibliothek zur Analyse von Stelleninseraten

Projekt:	IP5
Autoren:	Kevin Kirn
	Hoang Tran
Klasse:	7Ibb
Ort, Datum:	Brugg, 22. Dezember 2017

Fachhochschule:	FHNW
Hochschule:	Hochschule für Technik
Studiengang:	Informatik
Betreuender Dozent:	Prof. Dr. Manfred Vogel
	Ivo Nussbaumer
Auftraggeber:	Michael Henninger

Zusammenfassung

Stellenangebote werden heute vorwiegend im Internet publiziert. Dieser Weg ist kostengünstig und erreicht ein breiteres Publikum als dies bei herkömmliche Print-Anzeigen möglich ist. Firmen, welche nach neuen Mitarbeiten Ausschau halten, können Ihre Inserate so innerhalb von wenigen Minuten publizieren. Für Stellensuchende ist die resultierende Auswahl enorm gross. Aus tausenden Inseraten wollen die am besten passenden herausgefiltert werden. Um eine möglichst genaue Suche anbieten zu können, ist es heute unerlässlich, dass Eigenschaften der Inserate so genau und komplett wie möglich erfasst werden. Zudem möchten Interessenten nach möglichst vielen und spezifischen Attributen eingrenzen können. Eine automatische Analyse von eingereichten Inseraten würde den manuellen Aufwand minimieren, ebenso würde die Qualität sowie die Auffindbarkeit der Inserate maximiert.

Im Umfang des vorliegenden Projekts «Job Annotations» wurde eine Bibliothek entwickelt, mit welcher Stellenausschreibungen analysiert und annotiert werden. Annotiert bedeutet im konkreten Fall, dass Attribute wie beispielsweise die Stellenbezeichnung extrahiert und dem Inserat zugeordnet werden. Die resultierende Ausgabe enthält daher spezifische Informationen zu einem Inserat, welche zur weiteren Verarbeitung oder Filterung verwendet werden können.

In dem vorliegenden Bericht wird erläutert, wie die Umsetzung des Projekts erfolgte. Weiter werden Problemstellungen analysiert, Lösungsvorschläge und deren Umsetzungen dokumentiert sowie das schlussendlich erarbeitete Produkt näher beschrieben.

Inhaltsverzeichnis

1. Einführung

1.1. Ausgangslage

Mit der digitalen Revolution hat sich auch die Art und Weise geändert wie Firmen nach neuen Mitarbeiten suchen. Heute wird im Recruiting-Prozess ein grosses Augenmerk auf das Internet gelegt. Firmen inserieren ihre Job-Angebote nicht mehr in Zeitungen, sondern vorwiegend online, auf gut besuchten Stellenportalen sowie Vermittlungsbörsen. Durch diese Entwicklung steht stellensuchenden Personen eine immens grössere Auswahl und Übersicht der freien Stellen zur Verfügung. Inserate lassen sich nach Wünschen und Ansprüchen filtern. Die passende Stelle ist somit schneller identifiziert.

Für diese Filterung ist es jedoch nötig, dass Inserate korrekt und möglichst ausführlich erfasst werden. Für das Projekt IP5 der Fachhochschule Nordwestschweiz, im Auftrag von Herr Michael Henninger, wurde eine Bibliothek entwickelt, welche Inserate automatisiert analysieren kann. Mittels künstlicher Intelligenz werden Attribute der Inserate erkannt und somit filterbar gemacht. Dies ermöglicht eine verbesserte Filterung sowie eine Vielzahl an weiteren Möglichkeiten im Umgang mit Stelleninseraten.

1.2. Problemstellung

Die Schwierigkeit dieses Projekts besteht darin, dass Inserate analysiert werden müssen, welche nicht einer vordefinierten Norm folgen. Für das Verfassen von Stelleninseraten existieren Vorlagen sowie Erfolgsrezepte. Da Inserate jedoch von den verschiedensten Mitarbeitern einer Unternehmung verfasst werden können, ist die Vielfalt an Strukturen und Vorgehensweisen dennoch sehr gross.

Ziel dieses Projekts ist es spezifische Merkmale eines Inserates zu identifizieren. Dies kann beispielsweise die angegebene Stellenbezeichnung sein. Um die Aufmerksamkeit eines Interessenten zu schärfen, werden grade in Stellenbezeichnungen oft aber weitere Informationen eingebaut. So sind Titel bzw. Bezeichnungen wie «Erfahrener Softwareentwickler in Brugg mit Bezug zum Bankenwesen» keine Seltenheit. Die Richtigen Informationen aus solchen Inseraten zu extrahieren ist die Herausforderung.

1.3. Struktur und Leserführung

Dieser Bericht beschreibt die Erarbeitung des Informatik Projekts IP5 «Job Annotations» an der Fachhochschule Nordwestschweiz. Die Dokumentation ist in einen theoretischen Teil sowie einen praktischen Teil unterteilt. Im Ersten werden Problemstellungen und mögliche Lösungen diskutiert. Der praktische Teil dokumentiert die eigentliche Umsetzung der erarbeiteten Bibliothek.

2. Resultat

Folgend werden relevante Ergebnisse präsentiert, welche im Rahmen dieser Arbeit entstanden sind.

2.1. Beispiel Anwendung

Im Rahmen der vorliegenden Projektarbeit wurde eine Java-Bibliothek entwickelt. Ziel dieser ist es, Informationen aus Stellenausschreibungen zu extrahierten. Die Bibliothek ist alleine nicht lauffähig, sie kann jedoch in einem Projekt eingesetzt werden. Zur Illustrierung der Resultate wurde eine einfache Java-Anwendung entwickelt, welche einen Link zu einer Stellenausschreibung entgegennimmt. Dieser wird der Bibliothek übergeben, welche entsprechend die identifizierten Informationen zurückgibt.

Eingabe einer Stellenausschreibung

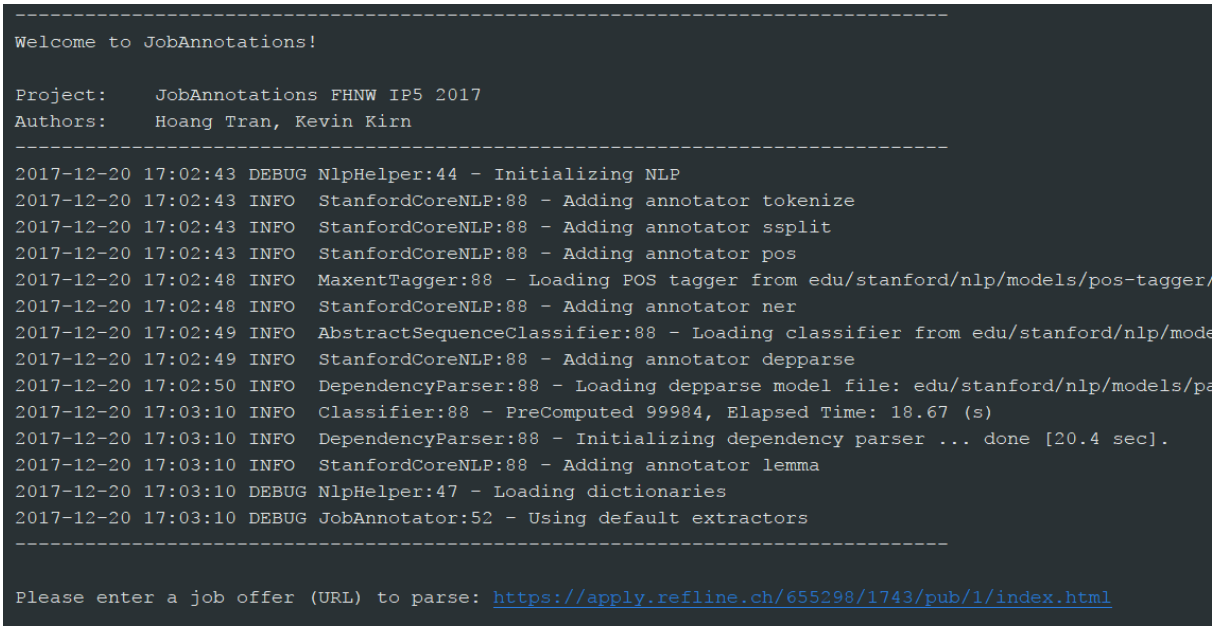

Abbildung 1 Screenshot der Beispiel-Anwendung

Die Anwendung wird während der Analyse verschiedenste Meldungen ausgeben, dies um eventuelle Fehler oder Beobachtungen besser verstehen zu können. Wie später in diesem Dokument beschrieben, lässt sich die Bibliothek einfach konfigurieren - so ist es unter anderem möglich diese Ausgaben zu verstecken. Nachdem die Bibliothek das Inserat verarbeitet hat, gibt diese wie erwähnt das Resultat an die Anwendung zurück. Diese stellt die Ergebnisse wie in folgender Abbildung gezeigt dar.

Resultat einer Analyse

```
--------------------------------------------------------------------------------
RESULT REPORT
--------------------------------------------------------------------------------
OrganisationExtractor:
Fachhochschule Nordwestschweiz FHNW

LanguageExtractor:
Deutsch, Englisch

WorkloadExtractor:
60 %

TitleExtractor:
JavaScript-Entwickler/in (60 )

LocationExtractor:
Windisch (AG)

SkillExtractor:
Design,OO,Administration,Issue,Serverinfrastrukturen,Liebe,Grundverständnis,Patterns


--------------------------------------------------------------------------------

Please enter a job offer (URL) to parse:
```

Abbildung 2 Resultat einer Analyse

Nachdem das Stelleninserat erfolgreich analysiert wurde, besteht die Möglichkeit ein nächstes Inserat anzugeben. So wurde eine Möglichkeit geschaffen, wie die erarbeitete Bibliothek auf einfache Art und Weise getestet werden kann. Die beschriebene Applikation wird gemeinsam mit der Bibliothek an den Kunden übergeben, so wird diesem ein konkretes Anwendungsbeispiel mitgeliefert.

Gesammelte Daten-Modelle

Wir haben die oben beschriebene Beispiel-Applikation bereits während der Entwicklungsphase rege benutzt. Als Resultat wurden zahlreiche Informationen von Stellenausschreibungen extrahiert. Diese Daten wurden nach sorgfältiger Prüfung in Dateien gespeichert, welche von der Bibliothek bei späteren Analysen wiederverwendet werden. So sind z.B. rund 2'500 Stellenbezeichnungen, 1’000 positive Skills, 1'500 negative Skills, 3'500 Namen von Unternehmungen sowie 116 Sprachen Teil der Daten in der Beispiel-Applikation. Mit diesen Daten entfällt eine notwendige Anlernphase der Anwendung. Diese Datensammlung, als Basis zur Analyse, wird ebenfalls an den Kunden weitergegeben.

2.2. Erfolgsquoten

Durch die Lernfähigkeit soll die Treffergenauigkeit verbessert werden. Die Lernfähigkeit wurde so implementiert, dass diverse Listen geführt werden. Diese Listen enthalten bekannte Begriffe, welche zur Laufzeit als «Dictionaries» geladen werden. Während der Extraktion der gewünschten Informationen, werden die gefundenen Begriffe mit den «Dictionaries» abgeglichen und die Bewertung entsprechend angepasst. Je grösser die Liste ist, desto besser die Genauigkeit der Extraktion. Die Listen sind als Textdateien abgespeichert und können jederzeit erweitert werden.

Zur Überprüfung der Qualitätssteigerung durch das Erlernen bekannter Begriffe wurden die Erfolgsquoten berechnet. Die folgende Tabelle zeigt die Treffergenauigkeiten der Extraktoren, als die Listen noch leer waren sowie die Treffergenauigkeiten nachdem Listen erweitert wurden. Es wurden jeweils 100 verschiedene Inserate analysiert. Die folgenden Werte wurden zur Illustrierung gerundet.

	Stellenbezeichnung	**Unternehmen**	**Standort**	**Pensum**	**Sprachkenntnisse**	**Benötigte Fähigkeiten**
Vorher 04.12.17	60%	50%	60%	80%	70%	45%
Nachher 20.12.17	95%	80%	80%	90%	95%	80%
Gewinn	35%	30%	20%	10%	25%	35%

Dabei wurde erfasst, ob es sich bei den extrahierten Informationen, um die gewünschten Informationen handeln. Daraus wurden die prozentuale Genauigkeit pro Extraktor berechnet. Wurden also von 100 analysierten Inseraten die Stellenbezeichnung von 40 Inseraten falsch oder nicht erkannt, entspricht die Erfolgsquote 60%.

Beim Extraktor der Fähigkeiten wurde ein anderes Verfahren verwendet. Hierbei wurde gemessen, wie hoch der Anteil der korrekten Fähigkeiten in Relation zu allen gefundenen Fähigkeiten ist. Würden in einem Inserat beispielsweise 10 Fähigkeiten erkannt werden, wobei drei davon keine Fähigkeiten sind, wäre die Erfolgsquote 70%. Aus allen analysierten Inseraten wurde der Mittelwert berechnet.

3. Analyse

Folgend werden während dem Projekt bearbeitete Problemstellungen sowie deren Lösungsansätze beschrieben. Ebenso werden im Vorfeld durchgeführte Abklärungen festgehalten.

3.1. Struktur von Inseraten

Wie bereits in der Einführung erwähnt, folgen Stelleninserate in der Regel keiner übergeordneten Normierung, was eine maschinelle Verarbeitung natürlich ausserordentlich vereinfachen würde. Bei eingehender Betrachtung und Analyse einiger Stelleninserate auf der Plattform www.jobs.ch fiel auf, dass ein Grossteil der Inserate jedoch einer gewissen Form bzw. Strukturierung folgt. So enthalten jene Inserate öfters folgende Informationen in separaten Blöcken in aufgeführter Reihenfolge:

- Informationen zur Unternehmung
- Stellenbezeichnung / Pensum
- Aufgaben welche bei der Stelle anfallen
- Qualifikationen welche ein potentieller Bewerber erfüllen sollte
- Kontaktinformationen zur angebotenen Stelle

Visualisierung der beschriebenen Struktur anhand eines Beispiels

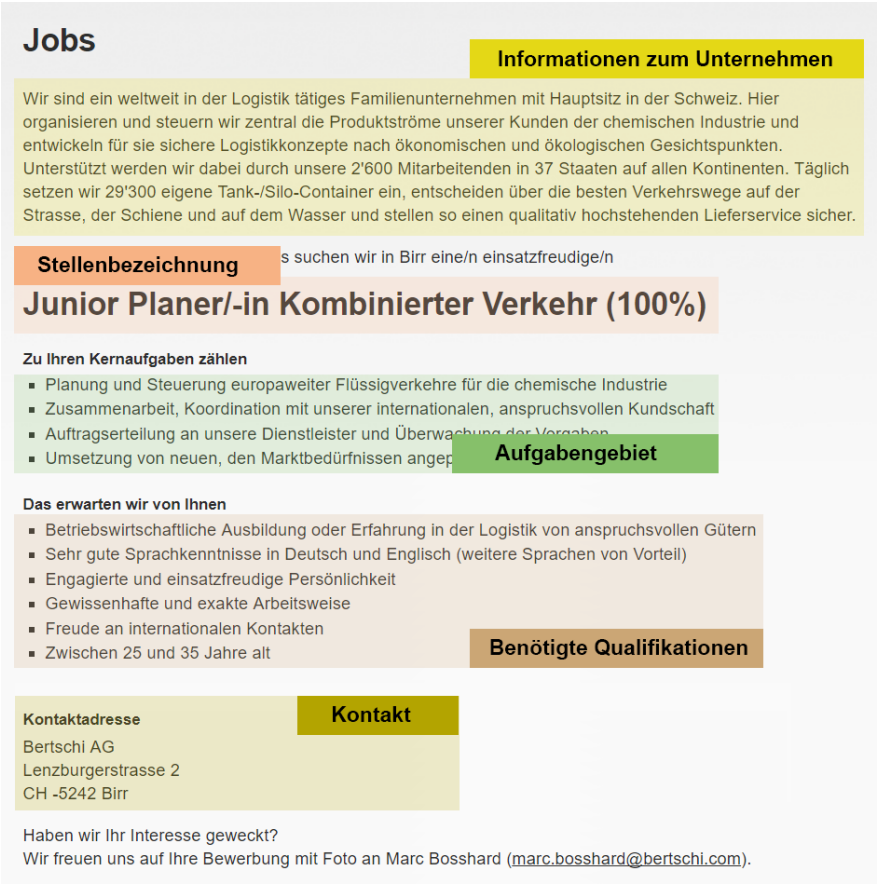

Abbildung 3 Stelleninserat der Firma Bertschi AG

Untersuchung von Stelleninseraten

Wie im obigen Abschnitt erwähnt, wurden Stelleninserate der Plattform www.jobs.ch analysiert. Die Plattform unterteilt Stellen in 18 Branchen. Pro Branche wurden 6 Inserate auf oben genannte Form bzw. Struktur untersucht. Total wurden somit 108 Stelleninserate analysiert, die Auswahl erfolgte zufällig. Es wurde jedoch darauf geachtet, dass pro Firma jeweils nur ein Inserat analysiert wurde, da diese oft gleich strukturiert sind. Von den untersuchten Inseraten wiesen 72% eine ähnliche bzw. die oben genannte Struktur auf. 28% der Inserate waren leicht oder komplett anders strukturiert.

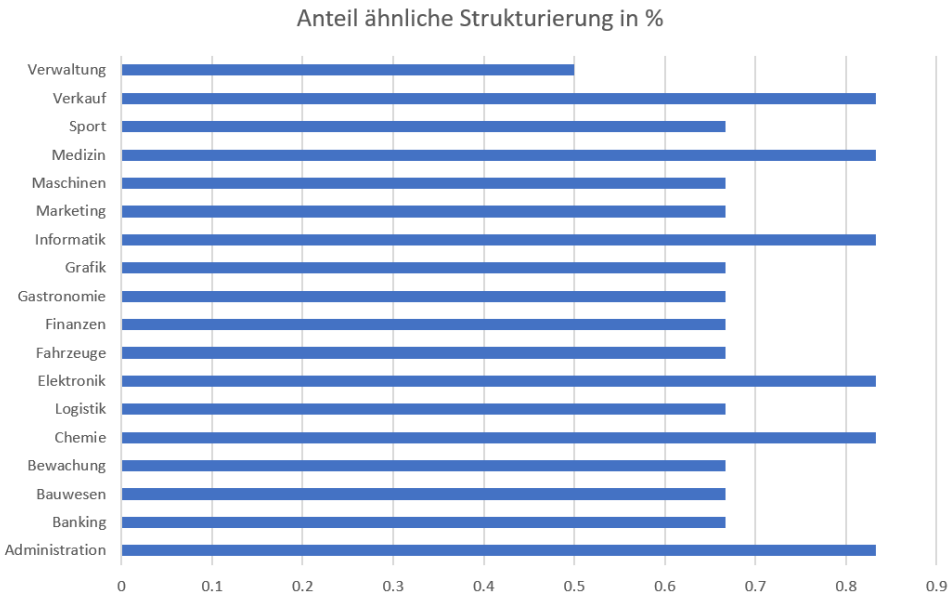

Abbildung 4 Ähnliche Strukturierung von Inseraten auf www.jobs.ch

Problemstellung

Eine gesuchte Information kann in einem Stelleninserat nur höchst selten bis nie eindeutig identifiziert werden. In den meisten Fällen ist das Resultat eine Annahme, die auf Wahrscheinlichkeiten basiert. Den Namen einer Firma zu identifizieren ist bereits schwierig, aber dann noch zu bestimmen, ob dieser auch wirklich der qualifizierte Ansprechpartner für einen Bewerber ist, stellt die richtige Herausforderung dar.

Lösungsansatz

Wie die obige Problemstellung zeigt, lässt sich oft nicht eindeutig bestimmen, ob es sich bei einer gefundenen Information auch tatsächlich um die relevante Eigenschaft handelt. Werden mehrere Informationen zu einem gesuchten Attribut gefunden, wie z.B. dem Namen der Firma, wird zusätzlich die Position bzw. der Kontext basierend auf obigen Feststellungen analysiert. Durch diesen Mehraufwand lässt sich die Wahrscheinlichkeit erhöhen, die relevante Information zu identifizieren.

3.2. Extraktion von Inhalten

Im vorliegenden Projekt sollen spezifische Informationen zu einem Stelleninserat extrahiert werden. Der Kunde wünscht, dass mindestens folgende Attribute eines online Inserates identifiziert werden:

- Stellenbezeichnung (auch Job-Titel genannt)
- Pensum der Anstellung
- Firmenname des Arbeitgebers
- Standort der Arbeitsstelle
- Benötigte Sprachkenntnisse
- Benötigte «Skills» - also Fähigkeiten des Bewerbers

Problemstellung

Das Erscheinungsbild eines Stelleninserates unterscheidet sich in der Praxis von Unternehmen zu Unternehmen äusserst stark. Die im Kapitel «3.1 Struktur von Inseraten» beschriebene Struktur, kann bei der Eruierung der Inhalte nur als Hilfestellung herbeigezogen werden. Für eine eindeutige Identifizierung der gesuchten Inhalte reichen diese Hinweise jedoch noch nicht aus. Problematisch ist zudem, dass die gesuchten Informationen selbst, also z.B. die Stellenbezeichnung, nicht immer gleich strukturiert sind.

Lösungsansatz

Um ein gesuchtes Attribut in einem Stelleninserat zu finden, wird eine Annäherungs-Strategie eingesetzt. Durch das Anwenden und Kombinieren verschiedener Techniken sollen jene Informationen identifiziert werden, welche mit der höchsten Wahrscheinlichkeit der Gesuchten entsprechen. Folgende Abschnitte beschreiben die genannten Techniken und deren Funktionsweise genauer.

<u>Indikatoren</u>

Ein vergleichsweise einfaches jedoch sehr wirksames Hilfsmittel sind Indikatoren, also Indizien, welche auf eine gesuchte Information hindeuten. Für die Analyse der Stelleninserate werden grundsätzlich zwei Arten von Indikatoren eingesetzt, dies sind textbasierte sowie visuelle Indizien. Textbasierte Indizien sind solche, welche oft in Verbindung mit einer Information verwendet werden. Nehmen wir den Job-Titel «Software Engineer m/w» als Beispiel. Das Kürzel «m/w» wäre in diesem Fall ein starker Indikator für einen Job-Titel, da diese in der Regel geschlechtsneutral ausgeschrieben werden.

Der Verfasser eines Stelleninserates legt üblicherweise einen gewissen Wert darauf, dass wichtige Informationen der Stelle schnell zu erkennen sind. So werden Stellenbezeichnungen, wie im Kapitel «3.1 Struktur von Inseraten» illustriert, durch Verwendung anderer Grösse, Farbe oder Art der Schrift visuell hervorgehoben, um dem Leser Relevanz zu signalisieren. Bei der Analyse der Stelleninserate wird die visuelle Erscheinung einer Information also auch bewertet. Ist das obige Beispiel des «Software Engineer m/w» z.B. visuell gewichtet, ist dies ein zusätzlicher Indikator dafür, dass es sich tatsächlich um einen Job-Titel handelt. Dies sind somit visuelle Indikatoren.

Reguläre Ausdrücke

Manche Informationen, wie z.B. die Stellenprozente (Pensum), werden in der Regel gleich gegliedert. So gibt es für Pensen nur sehr selten Formate, die nicht dem Format «Prozentwert %» entsprechen. Tritt eine solche Kombination in einem Stelleninserat auf, also z.B. «60 %», so ist damit mit höchster Wahrscheinlichkeit das Pensum gemeint. Solch definierbare Formate lassen sich mittels regulären Ausdrücken beschreiben und durch Pattern Matching in Texten relativ einfach finden bzw. extrahieren.

Ungenaue Suche mittels Listen *(Fuzzy Search)*

Stelleninserate unterscheiden sich inhaltlich oft stark, grade in Bezug auf verschiedene Branchen. Gesuchte Informationen lassen sich zudem nicht immer über die oben genannten Methoden identifizieren. Als gutes Beispiel kann die Suche nach Qualifikationen bzw. Skills herbeigezogen werden. Über Indikatoren und reguläre Ausdrucke wird bestimmt, wo wahrscheinlich Skills niedergeschrieben sind. Nach der Entfernung uninteressanter Wörter, wie Verben oder Adjektiven (PoS), bleiben potentielle Skills-Kandidaten übrig. Um nun eine echte Qualifikation zu identifizieren, wird eine sogenannte ungenaue Suche gegen eine Liste bekannter Skills durchgeführt. Mittels diesem Verfahren werden nicht nur bekannte Skills gefunden, sondern auch solche welche deren bis zu einem gewissen Grad ähnlich sind. Umgekehrt lassen sich mittels gleichem Vorgehen auch Wörter identifizieren, die eher keine Skills sind, bzw. ähnlich zu einem Wort sind, welches als «keine Qualifikation» bekannt ist.

Werden zwei Wörter miteinander verglichen, wird die Distanz zwischen diesen wie folgt berechnet. Als Erstes wird untersucht, wie viele Zeichen gelöscht wurden. Zwischen «Hans» und «Han» wäre dies eine Distanz von 1. Danach wird untersucht, welche Zeichen ersetzt wurden. So würde die Distanz zwischen «Hans» und «Hand» ebenso 1 betragen. In dem nächsten Schritt wird analysiert welche Zeichen neu hinzugefügt wurden. Die Distanz zwischen «Hans» und «Hand!» wäre nun insgesamt 2, da 1 Zeichen ersetzt und 1 Zeichen hinzugefügt wurde. Im letzten Schritt wird berechnet, welche Zeichen

transportiert, also vertauscht wurden. Für jede gefundene Änderung wird jeweils ein Distanzpunkt addiert. Die resultierende totale Distanz zwischen zwei Zeichenketten zeigt numerisch auf, wie ähnlich sich zwei Wörter bzw. Zeichenketten sind.

Named Entity Recognition (NER)

Named Entity Recognition, kurz NER, ist eine Unteraufgabe aus dem Bereich der Informations-Extraktion. Mittels vorgefertigten Modellen können definierte Entitäten (z.B. Personen) in Texten identifiziert werden. Das Finden solcher Entitäten basiert nicht auf Listen, sondern viel mehr auf der Struktur und Grammatik in welcher die gesuchte Entität statistisch gesehen häufig auftritt. Werden mehrere Sprachen analysiert, ist es dementsprechend nötig sprachspezifische Modelle zu verwenden. Genannte Modelle werden mit Beispiel-Texten trainiert in welchen Gesuchtes speziell markiert wurde. Mit steigender Anzahl an unterschiedlichen Trainings-Daten resultieren qualitativ bessere Ergebnisse. Für die Analyse der Inserate wird NER eingesetzt, folgend eine Visualisierung von einem Beispiel.

Abbildung 5 NER Analyse auf https://spacy.io

Part of Speech Analysis (PoS)

Ein weiteres Hilfsmittel zur Extraktion von gesuchten Informationen ist die Analyse von Texten bzw. deren Satzstrukturierung. Zum einen kann zwischen Nomen, Adjektiven und Verben unterscheiden werden, was in vielen Fällen die Suche nach einer Information einschränkt (z.B. Extraktion von Ort). Weiter werden grammatikalische Abhängigkeiten unter den einzelnen Wörtern analysiert, wodurch Bedeutungen einzelner Textbausteine identifiziert bzw. verstanden werden können. Grade in der Untersuchung von möglichen Skills bietet die Analyse von Abhängigkeiten die Möglichkeit zu verstehen, ob ein Wort im Bezug zum angesprochenen Bewerber steht oder nicht. Die gewonnenen Ergebnisse fliessen in die Bewertung der Wahrscheinlichkeit mit ein und verbessern so die Resultate.

Folgende Grafik verdeutlicht die Abhängigkeiten welche unter Wörtern existieren.

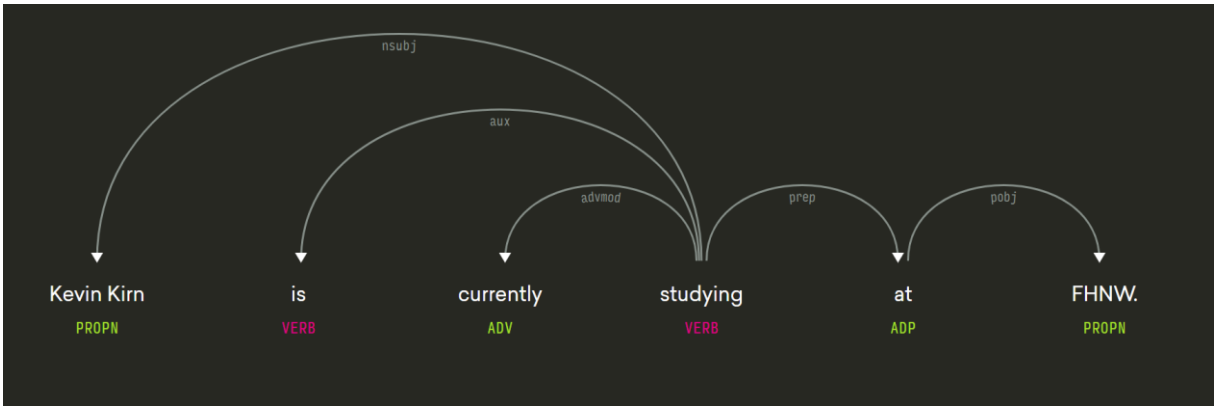

Abbildung 6 Part of Speech Abhängigkeiten Analyse auf https://spacy.io

3.3. Strukturelle Bewertung

Informationen welche in einem Inserat gesucht werden befinden sich an unterschiedlichen Positionen. Wie die Struktur im Kapitel «3.1 Struktur von Inseraten» illustriert, lassen sich z.B. Job-Titel öfters im oberen Bereich finden. Je nachdem welche Information gesucht wird, existiert eine Position an welcher diese statistisch gesehen öfters zu finden ist. Zusätzlich werden relevante Informationen des Inserats visuell hervorgehoben. Um die Qualität der Resultate weiter zu verbessern, sollen Informationen zusätzlich auf Grund ihrer Position und deren visueller Relevanz bewertet werden.

Problemstellung

Je nach dem um welche gesuchte Information es sich handelt, ist die optimale Position eine andere. Daher ist es nicht sinnvoll ein triviales Modell zu verwenden, welches annimmt, dass obere Informationen immer relevanter sind als weiter unten gefundene. Zeichenketten welche sich an guter Position finden lassen und zudem hervorgehoben sind, können dennoch eigentlich nicht relevant sein. Lediglich über die Positionierung lässt sich also eine Information nicht identifizieren. Die Position einer Information soll daher lediglich zu «Zusatzpunkten» zur Wahrscheinlichkeitsberechnung führen.

Lösungsansatz

Für jede gesuchte Information wird die optimale Position separat bewertet, diese bezieht sich auf die mehrfach genannte identifizierte Struktur von Inseraten. Ebenso wird jeweils definiert, ob eine visuelle Hervorhebung mit Zusatzpunkten bewertet wird. Um eine weitere Verbesserung in der Bewertung zu erreichen, wird zusätzlich auch gemessen, wie oft eine potentielle Information im Text vorkommt. Durch diesen zusätzlichen Aufwand soll verhindert werden, dass nur auf Grund von Struktur und visueller Erscheinung eine zu hohe Bewertung erfolgt. Folgend die Kriterien pro Information.

Gesuchte Information	**Position**	**Visuelle Relevanz**
Stellenbezeichnung	Höhere Position ergibt mehr Punkte	Wird bewertet
Pensum	Höhere Position ergibt mehr Punkte	Wird nicht bewertet
Art der Anstellung	Höhere Position ergibt mehr Punkte	Wird nicht bewertet
Firmenname	Tiefere Position ergibt mehr Punkte	Wird bewertet
Standort	Tiefere Position ergibt mehr Punkte	Wird nicht bewertet
Qualifikationen	Mittlere Position ergibt mehr Punkte	Wird bewertet
Sprachkenntnisse	Mittlere Position ergibt mehr Punkte	Wird nicht bewertet

3.4. Stellenbezeichnung

Stellenbezeichnungen sind elementarer Bestandteil eines Inserats. Es ist somit von wesentlichem Interesse, diese Information so zuverlässig wie möglich aus Inseraten zu extrahieren.

Problemstellung

Stellenbezeichnungen sind für potentielle Bewerber oft der erste Anhaltspunkt dafür, ob die ausgeschriebene Stelle ihrem Profil entspricht oder nicht. Die korrekte Identifikation, so dass diese im Kontext verständlich ist, ist eine Herausforderung. Die Varianz an Stellenbezeichnungen ist enorm, über alle möglichen Branchen müssen die korrekten Job Titel extrahiert werden. Zwar können Listen bekannter Stellenbezeichnungen geführt werden, gänzlich unbekannte Titel können dennoch nur schwer gefunden werden. Dass in Stellenausschreibungen Job Titel oft mit weiteren Information vermischt werden, erschwert die korrekte Identifikation zusätzlich. Es gilt also Wege zu finden, welche eine möglichst generische Identifizierung von Bezeichnungen ermöglicht. Dies unter Einhaltung sehr hoher Qualitätsansprüche an die Resultate.

Lösungsansatz

Die Text-Elemente der Job-Ausschreibung werden separiert und einzeln bewertet. Bei den Text-Elementen kann es sich um eine Überschrift, ein einzelnes fett geschriebenes Wort oder auch ganze Text-Abschnitte handeln. Die Bewertung wird an Hand von verschiedenen Bewertungskriterien angepasst, welche in den folgenden Abschnitten näher erläutert werden. Das Element, welches schlussendlich die höchste Bewertung hat, wird als Job-Titel klassifiziert.

Visueller Indikator

Im ersten Schritt werden die Text-Elemente basierend auf ihrer visuellen Signifikanz bewertet. Je stärker die Text-Elemente visuell hervorgehoben werden, desto höher ist die Bewertung.

Schreibweise-Indikator

Mit Schreibweise-Indikatoren sind typische Schreibweisen und Elemente gemeint, welche bei Stellenbezeichnungen oft zum Einsatz kommen.

Sie beinhalten die Endung «-/In» in Berufsbezeichnungen wie beispielsweise in «Sachbearbeiter/-In». Auch das Kürzel «m/w» wird berücksichtigt. Der letzte Schreibweise-Indikator ist das Pensum, da dies oft direkt in der Stellenbezeichnung steht. Je mehr Indikatoren ein Text-Element beinhaltet, desto höher ist seine Bewertung.

Sonderzeichen-Anteil

Die Bewertung wird basierend auf dem Anteil der Sonderzeichen im Text angepasst. Wenn der Anteil der Sonderzeichen im Text einen bestimmten Schwellenwert überschreitet, wird die Bewertung herabgesetzt.

Textlänge

Analog zum Sonderzeichen-Anteil, wird die Länge des Textes berücksichtigt. Übersteigt die Textlänge einen Schwellenwert, wird die Bewertung nach unten korrigiert.

Wiederholungen

Je öfter ein Text-Element in der Job-Ausschreibung vorkommt, desto besser wird dieses bewertet.

Abgleich mit Liste

Die Text-Elemente werden mit einer Liste von bekannten Job-Titeln abgeglichen. Die Bewertung eines Text-Elements wird dann verbessert, wenn es in der genannten Liste enthalten ist.

3.5. Unternehmen

Aus einem zu analysierenden Inserat soll der Name der Unternehmung extrahiert werden, welche die Stelle anbietet. Entsprechend ist der Arbeitgeber bzw. die zu kontaktierende Firma gesucht. Als Resultat soll möglichst der vollstände Firmenname mit Gesellschaftsform angegeben werden.

Problemstellung

Die Bezeichnung einer Firma ist bis auf einige Einschränkungen, zumindest in der Schweiz, frei wählbar. Dies führt dazu, dass eine enorme Vielfalt an möglichen Firmenbezeichnungen auftreten kann. Zählt ein Arbeitgeber Referenz-Kunden im Inserat auf, so können zudem mehrere Firmennamen auftreten. Wichtig ist daher, dass jene Firma identifiziert wird, welche durch Interessenten zu kontaktieren ist.

Lösungsansatz

Folgende Schritte werden bei der Analyse ausgeführt um die Firmenbezeichnungen zu identifizieren.

Gesellschaftsform Indikator

Die folgenden in der Schweiz zulässigen Abkürzungen für Gesellschaftsformen dienen als Indikator.

Abkürzungen *(in allen Landessprachen)*	**Gesellschaftsform**
AG, SA	Aktiengesellschaft
Gen, Genossenschaft, SCoop	Genossenschaft
GmbH, Sàrl, Sagl, Scrl	Gesellschaft mit beschränkter Haftung
KlG, SNC, SCl	Kollektiv
KmG, SCm, SAc	Kommanditgesellschaft
KmAG, SCmA, SAcA, SACm	Kommanditaktiengesellschaft

In den Inseraten wird nach oben genannten Abkürzungen gesucht, welche auf einen Firmennamen hindeuten. Sätze welche Abkürzungen enthalten, werden mit «PoS» auf deren Struktur untersucht. Mittels gewonnenen Struktur-Informationen werden Wörter links der Abkürzung gesucht, welche als Bestandteil des Firmennamens in Frage kommen. Konkret wird nach Nomen gesucht, welche links der Abkürzung stehen. Nachdem auf Verben oder Adjektive links der Abkürzung gestossen wurde, wird der Firmenname als komplett erkannt angesehen und zur weiteren Verarbeitung gespeichert.

Suche nach bekannten Unternehmen

Während der Analysephase wurden rund 3’500 Namen von Unternehmen auf der Job-Plattform www.jobs.ch gesammelt. Diese Liste wird verwendet um im Inserat nach bekannten Unternehmen zu suchen. Hierbei wird eine ungenaue Suche eingesetzt, so dass auch ähnliche Vorkommnisse als Firmenname identifiziert werden. Unbekannte Namen, welche von einer anderen Methode erkannt wurden, können ergänzt werden. Dies führt zu einer zunehmend wachsenden Liste an bekannten Unternehmen welche jeweils direkt erkannt werden können. Wird ein Name exakt über diese Methode gefunden, so wird die Bewertung höher ausfallen da es sich um eine bekannte Firma handelt.

Named Entity Recognition

Werden mit den vorhergehenden Methoden keine Unternehmen gefunden, so wird versucht mittels NER Firmen zu identifizieren. Hierfür wird ein Standard-Modell verwendet, welches in der deutschen Sprache Ergebnisse liefert. Die Qualität der Resultate ist höchst unterschiedlich, weshalb die obigen Methoden bevorzugt und bei Treffern höher bewertet werden.

Bewertung mittels Position

Im letzten Schritt werden die gefundenen Firmennamen anhand der jeweiligen Position sowie der Anzahl Vorkommnisse bewertet. Da Kontaktangaben in der Regel im unteren Bereich eines Inserats angegeben werden, wird für eine spätere Position eine höhere Bewertung vergeben. Nachdem die Bewertung vorgenommen wurde, wird der am besten bewertete Kandidat als Resultat ausgegeben.

3.6. Standort

Der Arbeitsort soll aus dem Inserat extrahiert werden. Hierbei ist die Stadt oder die Adresse gemeint, wo der Arbeitsnehmer bei der entsprechenden Stelle arbeiten würde.

Problemstellung

Der Arbeitsort wird üblicherweise nicht visuell hervorgehoben. Er wird oft irgendwo im Fliesstext erwähnt oder dann einfach als Kontaktadresse irgendwo ganz unten angegeben. Ein grosses Problem ist die korrekte Wahl, wenn mehrere Ortschaften in der Job-Ausschreibung vorkommen. Beispielsweise wird in der Job-Beschreibung der Standort des Firmen-Hauptsitzes erwähnt, welcher sich dann vom tatsächlichen Arbeitsort unterscheidet. Ein weiteres Problem ist, dass oft noch eine andere Adresse im Footer der Website steht, die nicht zum Inserat gehört, sondern eine Kontakt-Adresse des Portals ist.

Lösungsansatz

Wie bereits erwähnt werden oft im Footer der Website Adressen gefunden, die kein Bestandteil des Inserates sind. Deshalb werden potentielle Ortschaften, welche im Footer gefunden wurden, schlechter bewertet als Elemente, die sich im «Inhaltsbereich» der Website befinden.

Indikator

Im ersten Schritt werden Elemente gesucht, die speziell als Ortschaft gekennzeichnet sind. Hierbei wird im Dokument nach Schlüsselwörter gesucht, die prinzipiell Synonyme zum Begriff «Ortschaft» sind, wie beispielsweise «Arbeitsort», «Standort», «Arbeitsregion», etc. Es wird sowohl im Fliesstext nach als Ortschaft gekennzeichneten Texten gesucht, wie beispielsweise, wenn im Text «Arbeitsort: Bern» steht, als auch im Quell-Code nach Elementen die aus programmatischen Gründen mit diesen Schlüsselwörtern gekennzeichnet sind. Deshalb werden auch die englischen Äquivalente der Schlüsselwörter verwendet, weil für Programmcode die Begriffe in der Regel in Englisch geschrieben sind. Dieser Indikator arbeitet mit Synonymen des Begriffs «Ortschaft». Dadurch ist bereits gegeben, dass es sich mit höchster Wahrscheinlichkeit um Ortschaften handelt. Daher werden auf diese Weise gefundene Elemente am höchsten bewertet.

Regulärer Ausdruck

Mit einem regulären Ausdruck werden Textstellen gesucht, die aus einer vier- oder fünfstelligen Zahl und einem anschliessenden Wort bestehen. Damit sollen Kombinationen aus Ortschaften und Postleitzahlen gefunden werden. Dabei werden dann aber die Postleitzahlen ignoriert und lediglich die potentiellen Ortschaften für die Weiterverarbeitung behalten. Da mit dem regulären Ausdruck nur gewährleistet werden kann, dass kontextunabhängige Kombinationen von Zahlen und einem Wort gefunden werden, werden leicht auch Textstellen erkannt werden, die keine Ortschaften sind, wie beispielsweise «1000 Mitarbeiter». Deshalb werden mit dieser Methode gefundene Textstellen weniger hoch gewertet als mit der vorher beschriebenen Methode.

Named Entity Recognition

Mit NER wird nochmals der ganze Text analysiert, um Ortschaften zu identifizieren. Hierfür wird ein Standard-Modell verwendet, welches in der deutschen Sprache Ergebnisse liefert. Die Qualität der Resultate ist höchst unterschiedlich, weshalb die obigen Methoden bevorzugt und bei Treffern höher bewertet werden.

Filterung basierend auf Anzahl Wörter

Nachdem die Suche potentieller Ortschaften abgeschlossen ist, wird anschliessend die Filterung und Anpassung der Bewertungen durchgeführt. Hierbei werden jene Kandidaten eliminiert, welche mehr Wörter enthalten als der definierte Schwellenwert zulässt.

Validierung

Als Standort klassifizierte Informationen werden mit Hilfe einer Schnittstelle validiert, welche durch den Bund angeboten und betrieben wird. Mit Hilfe dieser wird geprüft, ob es sich bei einer Information um eine existierende Ortschaft handelt. Wenn das Resultat der Schnittstelle den Kandidaten entspricht, bzw. ähnlich ist, wird eine bessere Bewertung vergeben. Analog wird die Bewertung schlechter ausfallen, sollten sich Resultat und Kandidat zu sehr unterscheiden.

Duplikate

Abschliessend werden die Bewertungen angepasst, basierend darauf, wie oft eine Ortschaft gefunden wurde. Zur Bereinigung der Resultate werden alle Duplikate aus der Kandidaten-Liste entfernt. Hierbei wird jeweils der Eintrag in der Liste belassen, welcher am besten bewertet wurde.

3.7. Pensum

Das Pensum einer Arbeitsstelle soll aus der Job-Ausschreibung extrahiert werden.

Problemstellung

Das Pensum wird in der Regel nicht explizit als solches markiert. Typischerweise wird das Pensum direkt in der Stellenbezeichnung erwähnt. Es ist aber auch möglich, dass dieses sonst wo im Inserat erwähnt wird. Die Problematik besteht nun darin, das korrekte Pensum unabhängig von dessen Position zu identifizieren. Prozentwerte mit anderen Bedeutungen sollen nicht als Pensum klassifiziert werden.

Lösungsansatz

Das Pensum wird, wie in den folgenden Abschnitten beschrieben, ermittelt und bewertet. Der gefundene Wert, welcher am Schluss die höchste Bewertung hat, wird als Resultat verwendet.

Regulärer Ausdruck

Mit einem regulären Ausdruck wird der gesamte Text des Inserats nach Zahlen mit anschliessendem Prozentzeichen abgesucht. Diese stellen die potentiellen Pensum-Kandidaten dar. Weitere Methoden für die Suche von Pensum-Textstellen werden nicht verwendet.

Validierung

In diesem Schritt werden die gefundenen Prozent-Werte überprüft. Ungültige Pensum-Werte werden hier entfernt. Werte mit Zahlen über 100 oder unter 1 werden entfernt. Pensen werden meist in 5%-Schritten angegeben. Werte, welche von dieser Annahme abweichen, werden schlechter bewertet.

Duplikate

Abschliessend werden die Bewertungen angepasst, basierend darauf, wie oft ein Pensum gefunden wurde. Zur Bereinigung der Resultate werden alle Duplikate aus der Kandidaten-Liste entfernt. Hierbei wird jeweils der Eintrag in der Liste belassen, welcher am besten bewertet wurde.

3.8. Benötigte Fähigkeiten

Die vom Arbeitgeber geforderten Fähigkeiten sollen aus dem zu analysierenden Inserat extrahiert werden. Als Resultat sollen nur die Schlüsselwörter extrahiert werden, nicht etwa ganze Sätze, wie es üblicherweise in den Inseraten erfasst ist. Hierbei gilt zu beachten, dass in den Inseraten oft der Aufgabenbereich, das Angebot des Arbeitgebers und das Profil des Bewerbers beschrieben wird. Auch zu beachten gilt, dass Schulabschlüsse nicht als Fähigkeit eingestuft werden sollen.

Problemstellung

Die erste Hürde ist die Erkennung des Text-Bereiches, welcher die Fähigkeiten beinhaltet. Wie erwähnt ist wichtig, dass nicht andere Stellen, wie etwa der Aufgabenbereich zur weiteren Verarbeitung verwendet wird. Das nächste Problem besteht darin, die Schlüsselwörter aus den Sätzen zu extrahieren und dann zu entscheiden, ob es sich tatsächlich um eine Fähigkeit handelt.

Lösungsansatz

Bei der Erkennung des korrekten Text-Bereiches, aus welchem es die Fähigkeiten zu extrahieren gilt, werden zwei Ansätze verfolgt. Beim Ersten, wird nach Listen gesucht, da die Fähigkeiten meistens als Auflistung erfasst werden. Bei der anderen Variante, wird durch Analyse des Textaufbaus versucht, einen Titel zu finden, welcher signalisiert, dass nun die Fähigkeiten beschrieben werden. So ein Titel wäre zum Beispiel «Ihr Profil».

Die identifizierten Text-Bereiche werden anschliessend danach bewertet, wie wahrscheinlich es ist, dass sie Fähigkeiten beinhalten. Weiter wird versucht die einzelnen Fähigkeiten aus den gefundenen Text-Bereichen zu extrahieren. Hierbei werden alle Nomina extrahiert und bewertet.

In den folgenden Abschnitten wird der Ablauf näher erläutert.

<u>Erkennung von Listen</u>

Im ersten Schritt wird das Dokument nach Listen-Elementen und den dazugehörigen Titel-Texten durchsucht. Es kann durchaus sein, dass eine Liste keinen Titel-Text hat. In diesem Fall, wird das Text-Element übernommen, welches sich direkt vor der Liste befindet. Dies kann ein einzelner Satz sein, aber auch ein ganzer Paragraph.

Nun erfolgt eine erste Bewertung der gefundenen Listen. Zunächst wird überprüft, ob eine Liste verdächtige Elemente beinhaltet. Damit sind vor allem Links oder Buttons gemeint. Dass sich solche Elemente in einer Auflistung von Fähigkeiten befinden, ist höchst unwahrscheinlich. Dies würde darauf hindeuten, dass es sich bei der gefundenen Liste um eine Navigation handelt. Deshalb gibt es für das Vorkommen dieser Elemente Abzüge bei der Bewertung.

Leider kommt es vor, dass Fähigkeiten zwar als Liste verfasst werden, jedoch nicht valide Listen-Elemente verwendet werden. Dabei werden beispielsweise «*»-Zeichen als Auflistungs-Punkte verwendet. Um auch solche Listen zu finden, wird der Text des Inserats nach Zeilen durchsucht, welche wiederholt mit dem gleichen Zeichen beginnen. Beim beginnenden Zeichen gilt es zu beachten, dass es sich um ein Zeichen handeln muss, dass nicht im deutschen Alphabet vorkommt. Wenn also eine Liste verfasst wird, dass als Auflistungs-Punk den Buchstaben «ü» verwendet, wird dies nicht erkannt. Zusätzlich muss ein Leerzeichen folgen. Jene Zeilen müssen eine Reihe bilden, welche durchaus Leerzeilen beinhalten darf. Wird eine solche Liste gefunden, wird der dazugehörige Titel-Text ermittelt.

Bewertung der Listen durch Indikatoren

Eine Auflistung von geforderten Fähigkeiten beschränkt sich in der Regel nicht auf einige wenige Punkte. Deshalb wird bei der Bewertung auch die Anzahl Elemente der Liste berücksichtigt. Falls sich nur drei oder sogar weniger Elemente in der Liste befinden, wird bei der Bewertung ein gewisser Wert abgezogen. Je weniger Elemente es sind, desto höher ist der abgezogener Wert.

Anschliessend wird der Anteil von Sonderzeichen in den Listen-Einträgen überprüft. Mit Sonderzeichen sind Zeichen gemeint, welche nicht im deutschen Alphabet (inklusive Umlaute) existieren. Davon ausgeschlossen sind Leerzeichen, Kommas, Punkte und Doppelpunkte. Macht der Anteil dieser Sonderzeichen mehr als einen Viertel des Textes eines Listeneintrags aus, gibt es Abzüge bei der Bewertung. Je grösser dieser Anteil, desto grösser der Abzug.

Nachfolgend wird der Titel analysiert. Auch hier wird überprüft, ob sich verdächtige Elemente, wie Links oder Buttons, im Titel-Element befinden. Dies hätte einen Bewertungs-Abzug zur Folge. Besteht der Titel-Text aus mehr als sechs Wörter, gibt es ebenso einen geringfügigen Abzug. Je mehr Wörter im Text vorhanden sind, desto schlechter wird die Bewertung ausfallen.

Bewertung der Listen mit PoS

Mittels «Part of Speech» werden die Titel-Texte analysiert. Dabei werden die grammatikalischen Abhängigkeiten zwischen den Wörtern ausgewertet. So können Wort-Kombinationen gefunden werden, welche auf geforderte Fähigkeiten hindeuten. Ebenso könnten diese Gruppen auf das Angebot des Arbeitgebers oder den Aufgabenbereich hindeuten.

Dabei werden Kombinationen von bestimmten Wort-Arten überprüft und mit verschiedenen Listen bekannter Wörter verglichen. Beispielsweise wird nach einem Personalpronomen sowie dem dazugehörigen Verb gesucht. Die beiden Wörter werden daraufhin mit den Indikatoren verglichen und entsprechend bewertet. Wenn es sich z.B. um die Kombination «Sie bieten» handelt, würde die Bewertung verbessert werden, da dies darauf hindeutet, dass dies ein Titel für eine Fähigkeiten-Liste ist.

Bei der Analyse werden die Titel-Texte auf drei Fälle überprüft. Der erste Fall ist, wie bereits erwähnt, dass der Titel-Text eine Kombination aus Personalpronomen und Verb beinhaltet. Der zweite Fall ist, dass der Titel-Text eine Kombination aus Possessivpronomen und Nomen beinhaltet. Der dritte und letzte Fall ist, dass der Titel-Text nur aus einem einzigen Nomen besteht. Diese drei Varianten werden in den folgenden Abschnitten näher beschrieben.

Bei der Überprüfung des ersten Falls wird die Bewertung wie folgt beeinflusst:

	Sie	**Wir**
Anbieten	Bewertung geht hoch	Bewertung geht runter
Erwarten	Bewertung geht runter	Bewertung geht hoch

Die Wörter werden dabei auch mit Wörtern verglichen, welche im Kontext die gleiche Bedeutung haben. Beispielsweise ist die Kombination «Sie bringen mit» gleich zu stellen wie «Sie bieten». Beide deuten auf eine Auflistung von Fähigkeiten hin.

Die folgende Tabelle zeigt einige Beispiele sowie deren Auswirkung auf die Bewertung:

Kombination	**Auswirkung**
Sie bringen mit	Bewertung geht hoch
Du bietest	Bewertung geht hoch
Sie erwartet	Bewertung geht runter
Wir verlangen	Bewertung geht runter
Ich erwarte	Bewertung geht runter
Wir bieten	Bewertung geht runter

Bei der Überprüfung des zweiten Falls wird pro Typ des Possessivpronomens unterschieden. Die Bewertungen fallen je nach Typ unterschiedlich aus.

Für die Variationen von «Ihre» (Ihr, deine, etc.) gilt folgendes Bewertungsschema:

Nomen	**Auswirkung**
Fähigkeit	Bewertung geht hoch
Arbeit	Bewertung geht runter
Möglichkeit	Bewertung geht runter

Beispiele dafür wären:

Kombination	**Auswirkung**
Dein Talent	Bewertung geht hoch
Ihre Qualifikationen	Bewertung geht hoch
Ihr Profil	Bewertung geht hoch
Ihre Tätigkeit	Bewertung geht runter
Deine Herausforderung	Bewertung geht runter
Ihre Perspektiven	Bewertung geht runter

Für die Variationen von «Unsere» (Unser, meine, etc.) gilt folgendes Bewertungsschema:

Nomen	**Auswirkung**
Erwartung	Bewertung geht hoch
Angebot	Bewertung geht runter

Beispiele dafür wären:

Kombination	**Auswirkung**
Unsere Erwartungen	Bewertung geht hoch
Meine Anforderungen	Bewertung geht hoch
Unser Angebot	Bewertung geht runter

Texte welche aus einem einzelnen Nomen bestehen werden wie folgt bewertet:

Nomen	**Auswirkung**
Fähigkeiten	Bewertung geht hoch
Anforderungen	Bewertung geht hoch
Arbeit	Bewertung geht runter
Angebot	Bewertung geht runter
Möglichkeiten	Bewertung geht runter

Beispiele dafür wären:

Nomen	**Auswirkung**
Qualifikationen	Bewertung geht hoch
Profil	Bewertung geht hoch
Anforderungen	Bewertung geht hoch
Herausforderung	Bewertung geht runter
Wirkungsfeld	Bewertung geht runter
Perspektiven	Bewertung geht runter

Filterung durch Bewertung

Nachdem die Listen bewertet wurden, werden jene entfernt, welche den Schwellenwert nicht erreicht haben. Falls keine Listen mehr vorhanden sind, wird versucht Fähigkeiten im Fliesstext zu finden.

Fähigkeiten-Suche mit PoS

Wie bereits erwähnt, wird dieser Schritt nur ausgeführt, falls durch Überprüfung der Listen keine Fähigkeiten gefunden wurden. In diesem Fall wird der gesamte Text des Inserats Zeile für Zeilen einzeln analysiert. Bei der Analyse der einzelnen Zeilen wird das selbe Verfahren verwendet, welches bereits im Abschnitt «Bewertung der Listen mit PoS» eingeführt wurde. So wird erkannt, ob die Zeile einen Titel für nachfolgende Fähigkeiten darstellt. Der nachfolgende Text wird für die Extraktion verwendet.

Extraktion der Fähigkeiten

In diesem Schritt wird versucht die Schlüsselwörter zu extrahieren, welche die geforderten Fähigkeiten darstellen. Entweder werden die gefundenen Listen oder Text-Blöcke verwendet.

Hierbei werden alle Nomina aus den Texten extrahiert und mit zwei Listen verglichen. Die erste Liste beinhaltet Wörter, die keine Fähigkeiten sind. Die zweite beinhaltet bekannte Fähigkeiten. Je nach Ähnlichkeit zu den Einträgen werden die Bewertungen angepasst. Falls ein Nomen einem identischen Eintrag in der Liste der Nicht-Fähigkeiten entspricht, wird dieser ignoriert. Falls nicht, werden Zusatzpunkte vergeben, sofern das Nomen ein Akronym bzw. eine Kombination aus Akronym und Nomen ist. Schliesslich wird aufgrund dieser Bewertung ein Nomen ignoriert oder als Fähigkeit klassifiziert.

3.9. Sprachkenntnisse

Arbeitgeber fordern von potentiellen Bewerbern sehr häufig Kenntnisse in verschiedenen Sprachen. Diese Anforderungen sollen aus einem Inserat extrahiert werden. Interessenten können durch diese Information sofort benötigte Sprachkenntnisse erkennen und deren Beherrschung beurteilen. Weiter soll auch das benötigte Niveau pro Sprache erkannt werden, sofern dieses angeben wurde.

Problemstellung

Eine Sprache, wie z.B. «Deutsch», in einem Text zu finden ist vergleichsweise einfach. Existierende Sprachen sind allgemein bekannt und in ihrer Anzahl limitiert. Die Schwierigkeit bei dieser Teilaufgabe besteht jedoch darin, jene zu identifizieren, welche an den Interessenten adressiert sind. Weiter ist das Erkennen der geforderten Niveaus je Sprache eine Herausforderung. Es muss identifiziert werden, welche Eigenschaften sich auf eine gefundene Sprache beziehen.

Beispiel: «*Sie haben sehr gute Englischkenntnisse und sprechen perfekt Deutsch.»*

In diesem Beispiel ist zu erkennen, dass Englisch sehr gut und Deutsch perfekt gesprochen werden muss, um den Anforderungen gerecht zu werden.

Lösungsansatz

Folgende Schritte werden bei der Analyse ausgeführt, um die Sprachkenntnisse zu identifizieren.

<u>Suche nach bekannten Sprachen</u>

Während der Analysephase wurde eine Liste der Schweizer Landessprachen sowie Englisch erstellt, jeweils in allen Übersetzungen der einzelnen Sprachen. Mit dieser Liste wird in einem Inserat eine ungenaue Suche ausgeführt, so dass auch andere bzw. ähnliche Kombinationen gefunden werden. Hiermit können Sätze identifiziert werden, in welchen Sprachen bzw. deren Kenntnisse genannt werden.

<u>Strukturelle Analyse der Sätze</u>

Die im ersten Schritt gefundenen Sätze werden mittels «PoS» auf deren Struktur untersucht. So werden Wörter identifiziert, welche sich auf die gefundene Sprache beziehen. Adjektive, welche die Sprache charakterisieren, werden extrahiert und als Niveau-Angabe interpretiert. Zusätzlich werden gefundene Wörter mit bekannten, bereits oft identifizierten verglichen.

3.10. Lernfähigkeit

Wie in den bisherigen Kapiteln ausführlich behandelt wurde, ist die Extraktion von Informationen aus Stelleninseraten meist nicht einfach. Die Aufgabe besteht darin, Eigenschaften zu identifizieren, welche für die Anwendung eigentlich unbekannt sind. Mittels unterschiedlichen Strategien (siehe detaillierte Ausführungen weiter oben) werden Kandidaten für gesuchte Informationen berechnet. Ob ein solcher Kandidat nun aber z.B. wirklich ein Skill ist, lässt sich nicht mit Sicherheit definieren. Es bleibt somit ein Restrisiko, dass die identifizierte Information falsch ist. Mittels maschinellem Lernen sollen gefundene Informationen «erlernt» und die Qualität der Resultate dadurch weiter verbessert werden.

Problemstellung

Die bei der Suche nach einer bestimmten Information gefundenen Kandidaten werden in der Regel untereinander bewertet (gemäss obigen Kapiteln). Der Kandidat mit der höchsten Bewertung wird als Resultat der Suche angesehen und somit zurückgegeben. Dieses Resultat soll nun der Anwendung langfristig bekannt gemacht werden, es soll also aus seinen Ergebnissen lernen. Hierbei treten im Wesentlichen zwei Probleme auf. Zum einen sollten im Idealfall nur wirklich korrekte Resultate angelernt werden. Würden falsche Resultate gelernt, so würden diese zu noch mehr falschen Annahmen führen. Weiter sollten die Daten möglichst nutzerfreundlich und portabel gespeichert werden, so dass händische Korrekturen der Lerndaten nicht verunmöglicht werden.

Lösungsansatz

Informationen, welche aus einem Stelleninserat extrahiert wurden, werden nicht direkt gelernt. Die Speicherung dieser Resultate erfolgt in eine Datei, welche unter einer Art Quarantäne steht. Dies erlaubt es dem Nutzer, die Resultate vorerst manuell gegen zu prüfen und wenn nötig zu korrigieren. Dem Anwender ist es danach möglich, die Quarantäne aufzuheben und die Daten effektiv in die gelernten zu integrieren. Hierdurch wird das eigentliche, selbständige maschinelle Lernen verhindert. Es stellt jedoch ebenso sicher, dass zu Beginn die Lerndaten nicht qualitativ minderwertig sind. Ist die Präzision der Analyse genügend hoch, könnte diese Schranke auch wieder deaktiviert werden.

Um es dem Benutzer möglichst leicht zu machen, werden wenn möglich alle gelernten Daten in Form von normalen Textdateien gespeichert. Somit wird sichergestellt, dass die Daten bearbeitet werden können. Ebenso ist somit die Portabilität der Bibliothek bzw. Anwendung gegeben.

4. Umsetzung

Folgend werden die technischen Konzepte, Herangehensweisen und Implementationen dokumentiert. Code wird präsentiert, sofern dieser von Nutzen ist. Als NER-Frameworks, welche in diesem Dokument erwähnt werden, wurden CoreNLP, OpenNLP sowie LingPipe verwendet.

4.1. Struktur der Bibliothek

Folgende Grafik illustriert die zentralen Komponenten der Bibliothek. Die Klasse «JobOffer» repräsentiert eine zu analysierende Stellenausschreibung und enthält alle wichtigen Informationen sowie Inhalte dieser. Mittels des Interfaces «IExtractor» wird definiert, welche Methoden eine beliebige Klasse implementieren muss, die zur Extraktion von spezifischen Inhalten dienen soll. In der zentralen Klasse «JobAnnotator» können Klassen registriert werden, welche dieses Interface implementieren. Dies erlaubt das einfache Austauschen und Erweitern der Bibliothek mit bestehenden oder neuen Extraktions-Klassen.

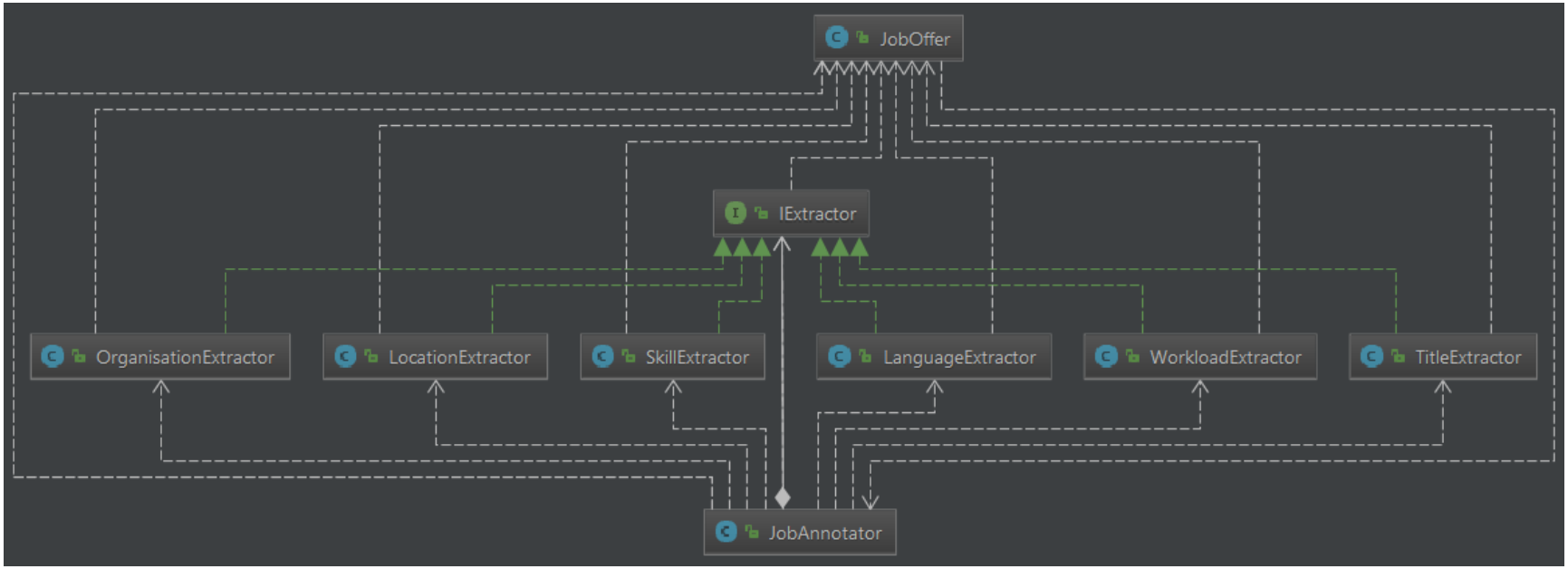

Abbildung 7 Ausschnitt aus Klassendiagramm

4.2. Konfiguration der Bibliothek

Da das Resultat dieses Projekts eine Java-Bibliothek ist, wurde nach einer einfachen Möglichkeit gesucht, wie diese bei Verwendung konfiguriert werden kann. Dem Endbenutzer soll es möglich sein, verwendete Model-Dateien oder Konfigurationen selbst zu definieren und diese einzusetzen. Dies ist elementar, da die Bibliothek nicht an die während der Entwicklung verwendeten Modelle gebunden sein soll. Gelöst wurde dies mittels Properties-Dateien, welche im Java-Umfeld weit verbreitet sind.

Konfigurations-Datei

Die Datei «job-annotations.properties» muss im Ressourcen-Ordner des Ziel-Projekts vorhanden sein. In dieser können für die Extraktoren und den eingesetzten NER-Bibliotheken (OpenNLP / CoreNLP) Pfade der Modell-Dateien definiert werden. Weiter ist es möglich, die Quarantäne beim Anlernen zu aktivieren bzw. zu deaktivieren – dies definiert, wo neu gefundene Resultate gespeichert werden.

```
# organisation extractor
extraction.organisations.train=./data/known_companies.txt

# language extractor
extraction.languages.train=./data/known_languages.txt

# skill extractor
extraction.skills.train.positive=./data/known_skills.txt
extraction.skills.train.negative=./data/known_anti_skills.txt

# title extractor
extraction.titles.train=./data/jobtitles.fhnw.raw

# location
extraction.locations.train=./data/known_locations.txt

# configuration of external libraries
external.StanfordCoreNLP.configuration=./configuration/StanfordCoreNLP-ger-
man.properties
external.OpenNLP.models.sentence=./open-nlp/de-sent.bin

# machine learning mode
configuration.ml.quarantine=true
```

ConfigurationUtil

Das Auslesen sowie die Verwendung der oben definierten Konfigurationen wird durch die Klasse «ConfigurationUtil» übernommen. Hierbei handelt es sich um eine Klasse, welche das Singleton-Pattern implementiert und somit bei Verwendung nur einmal Instanziiert wird. Dieses Pattern stellt sicher, das systemweit nicht verschiedene Konfigurationen verwendet werden, was zu Fehlern führen würde. Die

Datei wird nach Einlesen in eine Instanz der Klasse «java.util.Properties» (Standard-Bibliothek) abgebildet. Das konkrete Auslesen und halten der Konfigurationen ist wie folgt implementiert worden.

```
public final class ConfigurationUtil {

    private final static Logger LOG = Logger.getLogger(ConfigurationUtil.class);
    private final static String CONFIGURATION_FILE =
            "job-annotations.properties";

    private static ConfigurationUtil instance;
    private Properties properties;

    private ConfigurationUtil() {

        try {

            InputStream input =
                    FileUtils.getResourceInputStream(CONFIGURATION_FILE);
            properties = new Properties();
            properties.load(input);

        } catch (IOException e) {
            LOG.error("Failed to load " +
                    CONFIGURATION_FILE + ". Please see documentation.");
        }

    }

    private static ConfigurationUtil getInstance() {

        if (instance == null)
            instance = new ConfigurationUtil();

        return instance;
    }

    /**
     * Wrapper to grab properties from holder
     *
     * @param key the desired property name
     * @return the value of a found key
     */
    public static String get(String key) {
        return getInstance().properties.getProperty(key).trim();
    }

}
```

Eine Konfiguration wird in der Bibliothek nun wie folgt ausgelesen.

```
@Override
public void learn(String data) {

    String path = ConfigurationUtil.get("extraction.organisations.train");
```

4.3. Einlesen von Inseraten

Ein zu analysierendes Inserat wird mittels Hyperlink definiert. Dieser Link sollte, wenn immer möglich auf eine Detailseite eines Inserates zeigen, welche keine oder nur wenige Fremdelemente enthält. Links, welche viele Fremdelemente enthalten, werden unter Umständen schlechtere Resultate liefern.

Hyperlink Inhalte lesen

Um das Verarbeiten von Web-Inhalten zu vereinfachen, wird die Java-Bibliothek «Jsoup» in der aktuellen Version 1.10.2 verwendet. Diese liefert zu einem Link ein «Document», welches den DOM des Dokumentes hierarchisch repräsentiert. Mit dem so gewonnenen «Document» wird eine Instanz der Klasse «JobOffer» initialisiert, welche das Objekt für die weitere Verarbeitung normalisiert.

```
public HashMap<String, String> parse(String url) {

   HashMap<String, String> result = new HashMap<>();

    try {

        // create job offer by url
        Document document = Jsoup.connect(url).get();
        JobOffer jobOffer = new JobOffer(document);

        for (IExtractor extractor : extractors) {

            String candidates = extractor.parse(jobOffer);

            extractor.learn(candidates);

            result.put(extractor.getClass().getSimpleName(), candidates);

        }

    } catch (IOException e) {
        LOG.error("Something went wrong while parsing the job offer:\n", e);
    }

    return result;
}
```

Nachdem das gewünschte Dokument als «JobOffer» initialisiert wurde, wird für jeden registrierten Extraktor die parse-Methode aufgerufen, welche die «JobOffer» als Grundlage entgegennimmt. Die gewonnenen Resultate werden danach in eine HashMap abgelegt sowie nach Konfiguration angelernt.

Weitere Formen des Einlesens

Grundsätzlich ist die Bibliothek so implementiert, dass nicht zwingend eine Webseite als Basis dienen muss. Denkbare weitere Eingabeformate wären PDF-Dateien, Scans (OCR) oder Word-Dokumente. Im Rahmen dieser Arbeit wurde das Einlesen lediglich mittels Hyperlinks umgesetzt, für weitere Formate würde die Anpassung der Klasse «JobOffer» sowie einer entsprechenden Bibliothek jedoch genügen.

4.4. Extraktion von Inhalten

Es gibt verschiedenste Informationen, welche in einem Stelleninserat extrahiert werden können bzw. von Interesse sind. Da sich diese oft sehr voneinander unterscheiden, wird für jede Information ein sogenannter «Extractor» implementiert. Diese Klasse muss zwingend das Interface «IExtractor» implementieren, so dass die Erweiterbarkeit und Austauschbarkeit der Klassen in der Bibliothek gegeben ist.

Die Klasse «JobAnnotator» dient während der Verwendung als Hauptelement. Hier können Extraktoren registriert werden, welche ausgeführt werden sollen. Wird auf dem JobAnnotator die Methode parse aufgerufen, wird diese in den registrierten Klassen ausgeführt. Resultate der Klassen werden durch den «JobAnnotator» verwaltet und schlussendlich zurückgegeben.

```
public class JobAnnotator {

    private List<IExtractor> extractors;

    public JobAnnotator() { this(true); }

    public JobAnnotator(boolean useDefaultExtractors) {
        // load nlp models
        NlpHelper.getInstance();

        extractors = new ArrayList<>();

        if (useDefaultExtractors) {
            LOG.debug("Using default extractors");
            extractors.add(new TitleExtractor());
            extractors.add(new SkillExtractor());
            extractors.add(new LocationExtractor());
            extractors.add(new OrganisationExtractor());
            extractors.add(new LanguageExtractor());
            extractors.add(new WorkloadExtractor());
        }
    }

    public HashMap<String, String> parse(String url) {
        HashMap<String, String> result = new HashMap<>();

        try {
            // create job offer by url
            Document document = Jsoup.connect(url).get();
            JobOffer jobOffer = new JobOffer(document);

            for (IExtractor extractor : extractors) {

                String candidates = extractor.parse(jobOffer);

                extractor.learn(candidates);

                result.put(extractor.getClass().getSimpleName(), candidates);

            }

        } catch (IOException e) {
            LOG.error("Something went wrong while parsing the job offer:\n", e);
        }

        return result;
    }
...
```

4.5. Sammeln von Daten

Wir haben uns dazu entschieden, in den Extraktoren Daten von bereits identifizierten Informationen als Grundlage für die Analyse zu verwenden. Hierdurch können bekannte und ähnliche Daten klassifiziert werden. Durch das kontinuierliche Erweitern dieser Datensammlung soll die Qualität der Resultate stetig verbessert werden. Während der Entwicklung dieser Bibliothek mussten somit initiale Trainings-Daten gesammelt werden. Dies wurde mittels der Google-Chrome Erweiterung «WebScraper» durchgeführt. Mit dieser können automatisiert Informationen von Webseiten extrahiert werden.

Stellenbezeichnung

Als Grundlage wurden die Webseiten www.jobs.ch sowie www.ictjobs.ch verwendet und analysiert. Hierbei wurden rund 2'500 verschiedene Stellenbezeichnungen aus unterschiedlichen Branchen gesammelt. Diese wurden manuell bereinigt und in das initiale Job-Titel-Modell gespeichert. Zu Beginn wurde uns eine wesentlich umfangreichere Liste bereitgestellt, welche über 100'000 Einträge hatte. Diese Liste wurde zu beginn der Umsetzung verwendet, wies aber eine schlechte Qualität auf, welche sich nur schwer bereinigen lies. Da der «TitleExtractor» nicht nur auf bekannte Daten bei der Analyse setzt, identifiziert dieser mit der qualitativ hochwertigeren kleineren Liste ebenso gut wie vorher.

Fähigkeiten (Skills)

Um ein Gefühl dafür zu bekommen, was gängige Anforderungen sind, wurden rund 2'000 IT Stelleninserate der Plattform www.jobs.ch analysiert. Hierbei konnten rund 2'500 Wörter oder Wortgruppen extrahiert werden, welche potentielle Fähigkeiten darstellen. Für diese Daten wurde dann manuell beurteilt, ob es sich um eine Fähigkeit handelt oder nicht. Als Resultat wurde ein Modell erstellt, welches rund 900 identifizierte Skills enthält. Zusätzlich wurden jene Wörter in einem «Anti-Skill»-Modell erfasst, welche definitiv keine Fähigkeiten darstellen. Diese belaufen sich auf rund 1'700 Wörter.

Sprachen

Auf der Webseite www.weltsprachen.cc ist eine umfangreiche Sammlung an Sprachen dieser Welt vorhanden. Jene Liste wurde als Basis für das Sprachen-Modell verwendet. Zusätzlich wurden die Schweizer Landessprachen in Übersetzungen abgelegt, so dass ein Vorkommnis davon erkannt wird.

Unternehmen

Als Datengrundlage für den «OrganisationExtractor» wurden die 500 grössten Firmen der Schweiz im Modell erfasst (Quelle: http://www.segmentas.ch/top500). Weiter wurden rund 3'000 Firmennamen von der Webseite www.jobs.ch extrahiert und im Modell ergänzt.

4.6. Strukturelle Bewertung

Basierend auf den Annahmen, welche im Kapitel «3.3 Strukturelle Bewertung» getroffen wurden, wird die Struktur von Informationen bewertet. Um dieses Beispiel zu illustrieren, wird die Extraktion der Organisation zur Hilfe genommen.

Der Name einer Organisation ist z.B. meist ein Bestandteil der Kontaktadresse, welche sich in der Regel am Ende des Inserats befindet. Kandidaten, welche im unteren Teil des Dokuments klassifiziert werden, würden somit höher bewertet. Der folgende Code-Block ist ein Ausschnitt der Methode «parse» aus der Klasse «OrganisationExtractor».

```
...

int posScore = 0;
for (String candidate : legalFormCandidates) {

    if (StringUtils.isEmpty(candidate))
        continue;

    // check against found fuzzy search chunks
    for (Map.Entry<String, Integer> fuzzyCandidate :
            fuzzySearchCandidates.entrySet()) {

        if (candidate.toLowerCase().contains(
                fuzzyCandidate.getKey().toLowerCase())) {

            if (weightedIndicatorCandidates.containsKey(candidate)) {

                weightedIndicatorCandidates.put(candidate,
                        weightedIndicatorCandidates.get(candidate)
                                + 1 - fuzzyCandidate.getValue());

            } else {
                weightedIndicatorCandidates.put(candidate,
                        1 + posScore - fuzzyCandidate.getValue());
            }

        }

    }

    if (!weightedIndicatorCandidates.containsKey(candidate)) {
        weightedIndicatorCandidates.put(candidate, posScore);
    }

    posScore++;
}

...
```

Analoge Herangehensweisen werden in den anderen Extraktoren ebenfalls angewendet.

4.7. Extraktion Stellenbezeichnung

Bei der Extraktion der Stellenbezeichnung spielen die Tag-Namen der HTML-Elemente eine wichtige Rolle. Im ersten Schritt werden irrelevante HTML-Elemente entfernt.

```
for (String irrelevantTag : TitleExtractorConstants.IRRELEVANT_TAGS) {
    document.getElementsByTag(irrelevantTag).remove();
}
```

Bei den irrelevanten HTML-Elemente handelt es sich um diejenige, bei denen man davon ausgehen kann, dass diese die Stellenbezeichnung nicht beinhalten.

```
static final String[] IRRELEVANT_TAGS =
        {"style", "script", "meta", "link", "a", "input", "button"};
```

Danach wird der restliche HTML-Code der Website zeilenweise bewertet. Wenn eine Zeile ein bedeutsames HTML-Element beinhaltet, erhält die Zeile eine spezielle Bewertung. Mit bedeutsamen Elementen sind diejenigen gemeint, welche eine höhere oder tiefere Wahrscheinlichkeit für die Stellenbezeichnung aufweisen. Listenelemente erhalten eine Negativbewertung, weil es unwahrscheinlich ist, dass sich die Stellenbezeichnung in einer Liste befindet. Bei Elementen, welche visuell hervorgehoben werden, fällt die Bewertung je nach Stärke der visuellen Hervorhebung besser aus. Das «H1»-Element hat die höchste Bewertung. Basierend darauf werden die weiteren Bewertungen definiert.

```
static final IntStringPair[] RATINGS_HIGH_PRIORITY_TAG = {
        new IntStringPair(RATING_TAG_H1, "h1"),
        new IntStringPair(RATING_TAG_H1 / 4 * 3, "h2"),
        new IntStringPair(RATING_TAG_H1 / 2, "h3"),
        new IntStringPair(RATING_TAG_H1 / 4, "h4"),
        new IntStringPair(RATING_TAG_H1 / 4, "h5"),
        new IntStringPair(RATING_TAG_H1 / 4, "h6"),
        new IntStringPair(RATING_TAG_H1 / 4, "title"),
        new IntStringPair(RATING_TAG_H1 / 4, "b"),
        new IntStringPair(RATING_TAG_H1 / 4, "strong"),
        new IntStringPair(-RATING_TAG_H1, "li")
};
```

Alle anderen Zeilen, ohne speziell berücksichtigte Elemente, erhalten eine neutrale Bewertung mit dem Wert 0. Zeilen, welche nur HTML-Code beinhalten, aber sonst keinen Text, werden ignoriert.

Nun werden die initialen Bewertungen durch Überprüfung der Indikatoren angepasst. Bei den Indikatoren handelt es sich um die Angabe des Pensums, um die Geschlechtsneutralitätsformen wie beispielsweise die Abkürzung «w/m» oder z.B. das Suffix «In» in «LehrerIn» oder um eine Kombination aus diesen Indikatoren. Das Geschlechtsneutralitätsform-Suffix in der Stellenbezeichnung wird beibehalten. Übrige Indikatoren werden aus der Zeile entfernt, weil diese nicht Bestandteil des Job-Titels sind. Die Indikatoren werden mittels Verwendung von regulären Ausdrücken ermittelt. Im folgenden

Code-Block sind die regulären Ausdrücke der Indikatoren definiert. Zudem folgt ein Ausschnitt der Methode «extractRatedStringsFromHtml», welche die Logik zur Erkennung der Indikatoren und deren Bewertung enthält.

```
private static final String REGEX_GENDER_TEXT =
        "\\s?[wmfWMF]\\s?\\/\\s?[wmfWMF]\\s?";

private static final String REGEX_JOB_TITLE_GENDER_SUFFIX =
        "\\w[-\\/\\(\\|]+(in|IN|r|R)|\\w[-\\/\\(\\|]*In";

private static final String REGEX_WORKLOAD_GENDER_INDICATOR =
        "\\((" + WorkloadExtractor.WORKLOAD_REGEX + ")[^\\(\\)]*("
        + REGEX_GENDER_TEXT + ")\\)|\\((" + REGEX_GENDER_TEXT
        + ")[^\\(\\)]*(" + WorkloadExtractor.WORKLOAD_REGEX + ")\\)";

private static final String REGEX_GENDER_INDICATOR =
        "\\(?" + REGEX_GENDER_TEXT + "\\)?";

private static final String REGEX_WORKLOAD_INDICATOR =
        "\\(?" + WorkloadExtractor.WORKLOAD_REGEX + "\\)?";

static final IntStringPair[] JOB_TITLE_INDICATOR_REGEX_LIST = {
        new IntStringPair(0, REGEX_JOB_TITLE_GENDER_SUFFIX),
        new IntStringPair(1, REGEX_WORKLOAD_GENDER_INDICATOR),
        new IntStringPair(1, REGEX_GENDER_INDICATOR),
        new IntStringPair(1, REGEX_WORKLOAD_INDICATOR)};

...

for (IntStringPair tagRating :
        TitleExtractorConstants.RATINGS_HIGH_PRIORITY_TAG) {
    String tagName = tagRating.getString();
    String regex = String.format(TitleExtractorConstants.REGEX_FORMAT_TAG,
            tagName, tagName);
    Pattern pattern = Pattern.compile(regex);
    Matcher matcher = pattern.matcher(htmlLine);

    while (matcher.find()) {
        String matchedString = matcher.group();

        // remove all html tags
        matchedString = matchedString.replaceAll(
                TitleExtractorConstants.REGEX_TAG, "").trim();

        if (!matchedString.isEmpty()) {
            // use Jsoup to convert html special to normal chars (&amp; => &)
            matchedString = Jsoup.parse(Jsoup.parse(htmlLine).text()).text();

            // add extracted tag content and rating to list
            int rating = tagRating.getInt();
            IntStringPair ratedString =
                    new IntStringPair(rating, matchedString);
            extractedRatedStrings.add(ratedString);

            addedImportantTagContent = matchedString;
        }
    }
}

...
```

Die Bewertungen werden anschliessend nochmals angepasst, beispielsweise basierend auf der Anzahl der Sonderzeichen oder der Anzahl der Wiederholung des Begriffs im gesamten Dokument.

Am Schluss werden die bestbewerteten Kandidaten noch mit einem Dictionary von bekannten Job-Bezeichnungen abgeglichen. Wenn die Distanz der potentiellen Stellenbezeichnung mit einem Eintrag aus dem Dictionary klein genug ist, wird die Bewertung nochmals erhöht.

```
private void adjustRatingsByKnownJobTitleList(List<IntStringPair> ratedStrings){
    TrieDictionary<String> titlesDictionary =
            NlpHelper.getInstance().getTitlesDictionary();

    for (IntStringPair ratedString : ratedStrings) {
        String text = ratedString.getString();

        IntStringPair titleDistance = NlpHelper.getInstance()
                .calcDistanceWithDictionary(titlesDictionary, text, 1);

        if (titleDistance != null && titleDistance.getInt() == NlpHelper
                .DICTIONARY_DISTANCE_MIN_VALUE) {

            ratedString.setInt(ratedString.getInt()
                    + TitleExtractorConstants.RATING_DICTIONARY_MATCH);
        }
    }
}
```

Der Kandidat mit der besten Bewertung wird schlussendlich als Resultat zurückgegeben.

4.8. Extraktion Unternehmen

Durch die Kombination von drei unterschiedlichen Ansätzen wird versucht, Unternehmen in Stelleninseraten zu identifizieren. In einem ersten Schritt wird das Inserat nach jenen Indikatoren durchsucht, welche im Kapitel «3.5 Unternehmen» definiert wurden. Diese wurden wie folgt definiert.

```
// official legal form postfixes in switzerland
private static final String[] KNOWN_LEGAL_FORMS = {
        "AG", "Gen", "Genossenschaft", "GmbH", "KlG", "KmG", "KmAG", "SA",
"SCoop", "Sàrl",
        "SNC", "SCm", "SCmA", "Sagl", "SAc", "SAcA", "Scrl", "SCl", "SACm"
};
```

Ein positiver Fund stellt ein wesentliches Indiz dafür dar, dass es sich um eine Firma mit Angabe der Gesellschaftsform handelt. Wird in einem Satz eine Gesellschaftsform erkannt, so werden solange die Wörter links davon als Teil des Firmennamens dazugezählt, bis das nächste Wort nicht mehr vom Typ «COMMON_NOUN» oder «PROPER_NOUN» ist. Ausschnitt aus Methode «getLegalFormCandidates».

```
for (int i = tokens.size() - 1; i >= 0; i--) {

CoreLabel token = tokens.get(i);
    String word = token.get(CoreAnnotations.TextAnnotation.class);

    if (StringUtils.simplify(word).equalsIgnoreCase(legalForm)) {
        organisationNameBuilder = new StringBuilder();

    } else if (organisationNameBuilder != null) {

        String posTag = token.get(CoreAnnotations.PartOfSpeechAnnotation.class);

        boolean addToOrganisationName =
                        (posTag.equals(NlpHelper.POS_TAG_COMMON_NOUN) ||
                        posTag.equals(NlpHelper.POS_TAG_PROPER_NOUN));

        if (addToOrganisationName) {
```

In dem Fall, dass die Suche nach Indikatoren potentielle Unternehmungen als Resultat liefert, werden diese gegen bereits bekannte Unternehmungen verglichen. Dies geschieht mittels einer ungenauen Suche, so dass auch ähnliche Schreibweisen erkannt werden können. Anschliessend werden die potentiellen Unternehmungen nach Häufigkeit, Bekanntheit sowie deren Position bewertet.

```
if (candidate.toLowerCase()
        .contains(fuzzyCandidate.getKey().toLowerCase())) {

    if (weightedIndicatorCandidates.containsKey(candidate)) {

        weightedIndicatorCandidates.put(candidate,
                weightedIndicatorCandidates.get(candidate)
                        + 1 - fuzzyCandidate.getValue());

    } else {
        weightedIndicatorCandidates.put(candidate,
                1 + posScore - fuzzyCandidate.getValue());
    }
```

Wurde mit den obigen Herangehensweisen eine Unternehmung gefunden, welche zweifelsfrei und exakt anhand der bekannten Daten identifiziert werden konnte, wird davon ausgegangen, dass dies das bestmögliche Resultat darstellt. In diesem Fall wird der identifizierte Name als Resultat definiert.

```
long perfectMatches = fuzzySearchCandidates
        .entrySet().stream().filter(a -> a.getValue() == 0).count();

if (perfectMatches > 0) {

    String lastMatch = "";

    for (Map.Entry<String, Integer> entry : fuzzySearchCandidates.entrySet()) {

        if (entry.getValue() == 0) {
            lastMatch = entry.getKey();
        }

    }

    return lastMatch.trim();
}
```

Konnte hingegen keine bereits bekannte Firma erkannt werden, wird zur Sicherheit noch die dritte Herangehensweise angewendet. Hierbei wird mittels NER-Algorithmen (CoreNLP / Deutsche Models) nach Textfragmenten gesucht, welche mit «I-ORG» annotiert wurden. Dies bedeutet, dass das NLP-Framework diese Fragmente als Unternehmungen eingeschätzt bzw. klassifiziert hat.

```
for (CoreMap sentence : sentences) {

    List<CoreLabel> tokens = sentence
            .get(CoreAnnotations.TokensAnnotation.class);

    String currentOrganisation = "";

    for (int i = 0; i < tokens.size(); i++) {

        CoreLabel token = tokens.get(i);

        String word = token.get(CoreAnnotations.TextAnnotation.class);
        String ne = token.get(CoreAnnotations.NamedEntityTagAnnotation.class);

        if (ne.equals("I-ORG")) {
            currentOrganisation = currentOrganisation + " " + word;
        } else if (currentOrganisation != "") {
            candidates.add(currentOrganisation);

            LOG.debug("potential organisation found: " + currentOrganisation);

            currentOrganisation = "";
        }

    }

}
```

Nach einer erneuten, ähnlichen Bewertung, wird der am wahrscheinlichste Name zurückgegeben.

4.9. Extraktion Standort

Bei der Extraktion des Standorts wird unterschieden, ob ein potentieller Standort aus dem Footer der Website extrahiert worden ist oder nicht. Standorte aus dem Footer erhalten schlechtere Bewertungen. Im Footer befinden sich oft auch Kontaktadressen des Unternehmens. Bei Job-Portalen würde diese fälschlicherweise als Arbeitsort erkannt werden.

Zunächst werden Kandidaten mittels Indikatoren gesucht. Dabei werden zwei Ansätze verfolgt. Der erste Ansatz analysiert Begriffen, welche die Angaben zu Ortschaften signalisieren. Diese sind beispielsweise «Arbeitsort», «Standort», «Arbeitsregion», etc. Diese Begriffe werden in HTML-Attributen sowie dem Fliesstext gesucht. Beispielsweise könnte im Fliesstext «Arbeitsort: Bern» stehen, welcher dann erkannt werden würde. Im zweiten Ansatz wird ein regulärer Ausdruck verwendet. Damit wird nach einer Zeichenfolge gesucht, welche mit einer vier- bis fünfstelligen Zahl beginnt, gefolgt von einem Wort. Damit sollen Standortangaben gefunden werden, welche mit einer Postleitzahl gekennzeichnet sind, wie es bei der Angabe einer Kontaktadresse üblich ist.

```
static final String REGEX_ZIP_CODE = "\\d{4,5}\\s(.{2,})\\W";

...

private List<String> getPotentialJobLocationByZipCode(String plainText) {
    List<String> potentialLocations = new ArrayList<>();
    Matcher zipCodeMatcher = Pattern.compile(
            LocationExtractorConstants.REGEX_ZIP_CODE).matcher(plainText);
    while (zipCodeMatcher.find()) {
        potentialLocations.add(zipCodeMatcher.group(1));
    }
    return potentialLocations;
}
```

Anschliessend werden noch Standorte mit Hilfe von «Named Entity Recognition» extrahiert.

```
private List<String> getPotentialJobLocationByNamedEntityRecognition(
        Iterable<? extends CoreMap> annotatedSentences) {
    List<String> potentialJobLocations = new ArrayList<>();

    for (CoreMap sentence : annotatedSentences) {
        List<CoreLabel> tokens = sentence.get(
                CoreAnnotations.TokensAnnotation.class);
        for (CoreLabel token : tokens) {
            String word = token.get(CoreAnnotations.TextAnnotation.class);
            String nerTag = token.get(
                    CoreAnnotations.NamedEntityTagAnnotation.class);
            if (NlpHelper.NER_TAG_LOCATION.equals(nerTag)) {
                potentialJobLocations.add(word);
            }
        }
    }
    return potentialJobLocations;
}
```

Anschliessend werden die gefundenen Kandidaten bewertet. Berücksichtig wird die Bekanntheit eines Standortes, die Anzahl der Wiederholungen sowie die Textlänge. Zusätzlich werden die potentiellen

Arbeitsorte durch Verwendung einer Schnittstelle validiert. Der Standort wird mit dem Ergebnis abgeglichen. Abhängig von der Ähnlichkeit wird die Bewertung angepasst. Die Bewertung kann auch verschlechtert werden, wenn die von der Schnittstelle zurückgelieferte Adresse komplett anders ist. Folgender Code-Block enthält die Konstante, welche die URL-Vorlage für den Aufruf der Schnittstelle definiert. Zudem folgt ein Code-Ausschnitt der Methode «validateLocations», welche aufzeigt, wie ein einzelner Standort validiert und in dessen Bewertung angepasst wird.

```
static final String GEO_ADMIN_API_URL_TEMPLATE =
        "https://api3.geo.admin.ch/rest/services/api/SearchServer" +
                "?type=locations&limit=1&searchText=%s";

...

String location = ratedLocation.getString();
String validatedAddress = getValidatedAddressFromGeoApi(location);
if (validatedAddress != null) {
    addressValidated = true;
    int validationRating =
            calculateValidationRating(location, validatedAddress);
    ratedLocation.setString(validatedAddress);
    ratedLocation.setInt(ratedLocation.getInt() + validationRating);

} else {
    ratedLocation.setInt(ratedLocation.getInt()
            + LocationExtractorConstants.RATING_VALIDATION_FAILED);
}
```

Danach wird die Liste der potentiellen Arbeitsorte nach den Bewertungen sortiert. Einträge, welche nicht die höchste Bewertung aufweisen werden entfernt. Hat nur ein Eintrag die höchste Bewertung, bleibt dieser übrig. Dieser wird dann als Arbeitsort gewertet und zurückgegeben. Haben jedoch noch andere Einträge die gleiche Bewertung, bleiben diese auch übrig. In diesem Fall wird der Standort mit der kürzesten Textlänge als Resultat zurückgegeben.

4.10. Extraktion Pensum

Das Pensum einer ausgeschriebenen Arbeitsstelle lässt sich vergleichsweise einfach identifizieren. In den meisten Fällen, welche wir beobachtet haben, werden diese in der Form «[0-100]%» angegeben. Durch einen regulären Ausdruck wird nach solchen Formaten in den Inseraten gesucht.

```
public class WorkloadExtractor implements IExtractor {

final String WORKLOAD_REGEX = "(\\d+\\s*%?\\s*(.|\\w+)\\s*)?\\d+\\s*%";

...

List<IntStringPair> ratedWorkloads = new ArrayList<>();

Matcher workloadMatcher = Pattern.compile(WORKLOAD_REGEX).matcher(text);
while (workloadMatcher.find()) {
    String match = workloadMatcher.group();
    IntStringPair ratedWorkload = new IntStringPair(100, match);
    ratedWorkloads.add(ratedWorkload);
}
```

Wie im obigen Code-Block zu sehen ist, werden gefundene Pensen mit der Bewertung 100 versehen. In einem nächsten Schritt wird überprüft, ob diese valide sind bzw. sich im üblichen Rahmen bewegen.

```
private void validateWorkloads(List<IntStringPair> ratedWorkloads) {
    outer:
    for (int i = ratedWorkloads.size() - 1; i >= 0; i--) {
        IntStringPair ratedWorkload = ratedWorkloads.get(i);

        List<Integer> digits = new ArrayList<>();
        Matcher digitMatcher = Pattern.compile("\\d+")
                .matcher(ratedWorkload.getString());
        while (digitMatcher.find()) {
            digits.add(Integer.parseInt(digitMatcher.group()));
        }

        // remove workloads with invalid numbers of digits
        if (digits.size() < 1 || digits.size() > 3) {
            ratedWorkloads.remove(i);
            continue;
        }

        // remove workloads with invalid workload values
        for (Integer digit : digits) {
            if (digit < 1 || digit > 100) {
                ratedWorkloads.remove(i);
                continue outer;
            }
        }

        // decrease ratings of workloads with unlikely values
        for (Integer digit : digits) {
            if (digit % 5 != 0) {
                ratedWorkload.setInt(ratedWorkload
                        .getInt() + RATING_UNLIKELY_WORKLOAD);
            }

...
```

Nachdem Duplikate entfernt wurden, wird das bestbewertete Pensum als Resultat zurückgegeben.

4.11. Extraktion benötigter Fähigkeiten

Bei der Extraktion von Fähigkeiten werden zunächst Listen gesucht. Sowohl valide HTML-Listen als auch solche, welche im Fliesstext händisch erfasst wurden. Wie beispielsweise, wenn man «*» als Auflistungspunkt verwendet anstatt korrektes HTML zu benutzen. Für diese Listen wird jeweils der entsprechende Titel ermittelt. Dabei wird das letzte Text-Element verwendet, welches sich direkt vor der Liste befindet. Im besten Fall ist dies eine Zeile, wie z.B. «Ihre Fähigkeiten:». Es kann aber durchaus auch ein kompletter Absatz übernommen werden, falls dies das letzte Text-Element vor der Liste ist.

Die Listen-Titel erhalten jeweils eine Bewertung. Dabei werden die Wörter im Text auf ihren Typ und auf ihre grammatikalischen Abhängigkeiten überprüft. Dies geschieht durch Auslesen der Semantik, welche durch die NLP-Annotierung erkannt wird. Im folgenden Code-Block ist von der Methode «calcTitleDependencyRating» das erste if-Statement ersichtlich. Hier würde der Text «Ihre Fähigkeiten» im Zweig landen, wo als Kommentar «#Markierung» steht.

```
private int calcTitleDependencyRating(TypedDependency dependency) {
    String shortName = dependency.reln().getShortName();
    String word = dependency.dep().backingLabel().value().toLowerCase();

    IndexedWord gov = dependency.gov();
    String govTag = gov.tag();
    String govWord = gov.backingLabel().value().toLowerCase();

    if (EnglishGrammaticalRelations.DETERMINER.getShortName().equals(shortName)
            && "NN".equals(govTag)) {
        if (SkillExtractorConstants.DETERMINERS_YOUR.contains(word)) {
            if (wordContainsListItem(SkillExtractorConstants
                    .NOUNS_SKILL_SYNONYM, govWord)) {
                return 25; // #Markierung

            } else if (wordContainsListItem(SkillExtractorConstants
                    .NOUNS_JOB_SYNONYM, govWord)) {
                return -25;

            } else if (wordContainsListItem(SkillExtractorConstants
                    .NOUNS_POSSIBILITY_SYNONYM, govWord)) {
                return -25;
            }

        } else if (SkillExtractorConstants.DETERMINERS_OUR.contains(word)) {
            if (wordContainsListItem(SkillExtractorConstants
                    .NOUNS_EXPECTATION_SYNONYM, govWord)) {
                return 25;

            } else if (wordContainsListItem(SkillExtractorConstants
                    .NOUNS_OFFER_SYNONYM, govWord)) {
                return -25;
            }
        }
    }

...
```

Die gleiche Methode wird auch verwendet, um Fähigkeiten in Fliesstexten zu finden, welche nicht als Auflistung erfasst wurden. Dabei wird zeilenweise eine Bewertung eines Listen-Titels vorgenommen.

Zeilen mit mehr als sechs Wörter werden ignoriert. Ist die Bewertung positiv, werden die nachfolgenden Zeilen als Fähigkeiten klassifiziert, bis eine Leerzeile kommt. Folgender Code-Block zeigt die Implementation dieses Verfahrens.

```
private Map<IntStringPair, List<String>> extractSkillListsBySentenceAnalyzing(
        JobOffer jobOffer) {
    Map<IntStringPair, List<String>> skillLists = new HashMap<>();

    String[] lines = jobOffer.getPlainText().split("\n");
    IntStringPair lastTitle = null;
    boolean parsingSkills = false;
    for (String line : lines) {
        if (line.trim().isEmpty()) {
            if (parsingSkills) {
                List<String> skillList = skillLists.get(lastTitle);
                if (!skillList.isEmpty()) {
                    parsingSkills = false;
                }
            }
            continue;
        }

        if (parsingSkills) {
            List<String> skillList = skillLists.get(lastTitle);
            if (skillList != null) {
                skillList.add(line);
            }
        }

        int nofWords = line.split("\\s").length;
        if (nofWords > 6) {
            // ignore lines with many words
            continue;
        }

        int rating = calcSkillListTitleRating(jobOffer, line);
        if (rating > 0) {
            parsingSkills = true;
            lastTitle = new IntStringPair(0, line);
            skillLists.put(lastTitle, new ArrayList<>());
        }

    }
    return skillLists;
}
```

Die Fähigkeiten-Listen werden nicht nur anhand des Titels bewertet, sondern auch anhand der Listen-Einträge. Werden verdächtige HTML-Elemente, wie z.B. Buttons oder Links erkannt, würde die Liste eine schlechtere Bewertung erhalten. Ist die Bewertung der Listen abgeschlossen, werden alle Nomina aus den Sätzen extrahiert, so dass im Ergebnis nur noch Nomen vorhanden sind. Dabei werden zwei Dictionaries verwendet. Eines mit bekannten Skills, das andere mit bekannten Begriffen, welche wir Anti-Skills nennen. Der folgende Code-Block enthält den wichtigsten Teil der Methode «calcSkillNoun-Rating», welche die Bewertung des Begriffs übernimmt. Dabei wird auch jeweils eine vereinfachte Form des Begriffs mit den Dictionaries abgeglichen. Dies bedeutet, dass alle Buchstaben des Begriffs

in Kleinbuchstaben umgewandelt und Sonderzeichen entfernt werden. Die Distanzen werden anschliessend miteinander verglichen. Die Bewertung des Nomens wird daraus berechnet.

```
private int calcSkillNounRating(String noun) {
    String simplifiedNoun = StringUtils.simplify(noun);

    ...

    // combine distances
    int skillDistance = 0;
    if (skillWordDistance != null) {
        skillDistance += skillWordDistance.getInt();
    } else {
        skillDistance += SkillExtractorConstants
                .SKILL_NOUN_DEFAULT_DISTANCE_ADJUST_VALUE;
    }
    if (simplifiedSkillWordDistance != null) {
        skillDistance += simplifiedSkillWordDistance.getInt();
    } else {
        skillDistance += SkillExtractorConstants
                .SKILL_NOUN_DEFAULT_DISTANCE_ADJUST_VALUE;
    }

    int antiSkillDistance = 0;
    if (antiSkillWordDistance != null) {
        antiSkillDistance += antiSkillWordDistance.getInt();
    } else {
        antiSkillDistance += SkillExtractorConstants
                .SKILL_NOUN_DEFAULT_DISTANCE_ADJUST_VALUE;
    }
    if (simplifiedAntiSkillWordDistance != null) {
        antiSkillDistance += simplifiedAntiSkillWordDistance.getInt();
    } else {
        antiSkillDistance += SkillExtractorConstants
                .SKILL_NOUN_DEFAULT_DISTANCE_ADJUST_VALUE;
    }

    ...

    // check diff
    return antiSkillDistance - skillDistance;
}
```

Begriffe mit Negativbewertungen werden verworfen. Bewertungen mit dem Wert 0 sind neutral. Diese wurden weder in der Skill-Liste, noch in der Anti-Skill-Liste gefunden und werden nicht verworfen, da wir lieber etwas zu viel Fähigkeiten zurückliefern, als gewünschte Skills fälschlicherweise zu verwerfen.

4.12. Extraktion Sprachkenntnisse

In einer Modell-Datei haben wir alle Schweizer Landessprachen sowie Englisch erfasst, dies jeweils in allen Übersetzungen dieser Sprachen. Nach einigen Tests haben wir festgestellt, dass mittels dieser Liste bereits sehr gut benötigte Sprachkenntnisse identifiziert werden können. Hierzu wird im Inserat nach Wörtern gesucht, welche entweder direkt einer Sprache entsprechen oder diese enthalten.

```
public String parse(JobOffer jobOffer) {

    LOG.debug("Started to parse language from offer");

    // get the visible text from document
    List<String> lines = jobOffer.getPlainTextLines();

    Map<String, String> fuzzySearchCandidates =
            getKnownLanguagesCandidates(lines);

    // return comma separated list
    String result = "";

    for (Map.Entry<String, String> entry : fuzzySearchCandidates.entrySet()) {
        result += entry.getKey();

        if (entry.getValue() != null && entry.getValue() != "")
            result += "(" + entry.getValue() + "), ";
        else
            result += ", ";
    }

    if (result != "") {
        return result.substring(0, result.length() - 2);
    } else {
        LOG.debug("No languages found");

        return "";
    }
}

...

private Map<String, String> getKnownLanguagesCandidates(List<String> lines) {

    Map<String, String> candidates = new HashMap<>();

    // load known languages from model as list
    List<String> languages = FileUtils.getFileContentAsList(
            ConfigurationUtil.get("extraction.languages.train"));

    for (String language : languages) {

        for (String sentence : lines) {

            if (sentence.toLowerCase().contains(language.toLowerCase())) {
```

Das Niveau, mit welchem der Bewerber die Sprache beherrschen muss, soll pro Sprache identifiziert werden. Hierzu werden die Sätze, in welchen eine Sprache gefunden wurde, mittels POS annotiert. Der resultierende «SemanticGraph», welcher die Abhängigkeit der Wörter untereinander repräsentiert, wird dazu verwendet, Adjektive zu finden, welche sich auf die Sprache beziehen.

```
if (sentence.toLowerCase().contains(language.toLowerCase())) {

    // annotate sentence
    CoreMap annotatedSentence = NlpHelper
            .getInstance().getAnnotatedSentences(sentence).get(0);

    // load dependency graph from annotated sentence
    SemanticGraph dependencies =
            annotatedSentence.get(SemanticGraphCoreAnnotations
                    .CollapsedCCProcessedDependenciesAnnotation.class);

    // find word, which contains language
    IndexedWord languageWord = dependencies
            .getNodeByWordPattern("(?i)" + language + ".*");

    if (languageWord != null) {

        // find all words which are descendants of the language
        Set<IndexedWord> desc = dependencies.descendants(languageWord);

        // extract adjectives from dependencies
        List<String> predicates = desc.stream()
                .filter(d -> d.tag().equals("ADV") ||
                        d.tag().equals("ADJA") || d.tag().equals("ADJD"))
                .map(x -> x.word())
                .collect(Collectors.toList());

        // join found adjectives together
        String adjectives = String.join(" ", predicates);

        // save found language and adjectives
        candidates.put(language, adjectives);

    } else {

        // fall back with trivial contains
        if (sentence.toLowerCase().indexOf(language.toLowerCase()) > -1) {
            if (candidates.get(language) == null)
                candidates.put(language, "");
        }

    }

}
```

Die so identifizierten Sprachen werden mit den Adjektiven, welche das Niveau beschreiben, als Resultat zurückgegeben. Bei der Extraktion der Sprache haben wir uns bewusst darauf beschränkt, die Sprachen anhand einer Liste zu identifizieren. Es wäre in einem nächsten Schritt durchaus möglich, weitere Herangehensweisen wie z.B. «Named Entity Recognition» (NER) zu implementieren. Weiter gehen wir aktuell davon aus, dass das Niveau mittels Adjektiven definiert wird – was in sehr vielen Fällen auch bestens funktioniert. In manchen Fällen wird das Niveau anders umschrieben, jene können durchaus in einer Erweiterung implementiert werden, was zu besserer Erkennung des Niveaus führen könnte.

4.13. Lernfähigkeit

Wie in den oberen Abschnitten bereits erwähnt wurde, verwenden Extraktoren Modelle, welche bekannte bzw. identifizierte Entitäten enthalten. Dadurch können bekannte Informationen zweifelsfrei erkannt werden. Informationen welche zu einem bekannten Datensatz eine Ähnlichkeit aufweisen, können so besser oder erst dadurch als gesuchte Information klassifiziert werden. Die Resultate welche von Extraktoren identifiziert wurden, werden in den Modellen ergänzt. Hierdurch ergibt sich die eigentliche Lernfähigkeit der Bibliothek. Je mehr Daten verarbeitet und Informationen identifiziert werden, desto grösser wird der Bestand an Daten, welche bekannt sind. Jeder Extraktor verwaltet ein eigenes Modell, meist in Form von einfachen Textdateien, was die Wartbarkeit vereinfacht. Jeder Extractor implementiert eine Methode «learn». Sie definiert, wie die neuen Daten gespeichert werden.

Folgendes Beispiel illustriert das Anlernen neuer Daten anhand der Klasse «SkillExtractor». Neu erkannte Skills werden mittels der Klasse «FileUtils» in die konfigurierte Trainings-Datei gespeichert. Wie in der Konfiguration zu sehen ist, existiert ebenfalls eine negativ-Liste. Diese dient dazu Wörter bei der Analyse auszuschliessen, welche definitiv keine Anforderung darstellen. Diese Liste wird nicht durch die Bibliothek angelernt, sondern dient zur manuellen Verbesserung der Ergebnisse.

```
@Override
public void learn(String data) {
    data = data.replaceAll(SkillExtractorConstants.SEPARATOR, "\n");
    FileUtils.addDataToTrainFile(
            ConfigurationUtil.get("extraction.skills.train.positive"), data);
}

...

# skill extractor
extraction.skills.train.positive=./data/known_skills.txt
extraction.skills.train.negative=./data/known_anti_skills.txt
```

Das Speichern von neuen Daten wird dahingehend sichergestellt, dass Duplikate in bestehenden Modell-Dateien jeweils eliminiert werden. Folgend ein Ausschnitt aus der Klasse «FileUtils».

```
// make sure duplicates get ignored
HashSet<String> newContents = new HashSet<>();

// reader to train or train.quarantine file
BufferedReader br = new BufferedReader(new FileReader(filename));

// read first line to process
String line = br.readLine();

// read content into set
while (line != null) {
    newContents.add(line);

    line = br.readLine();
}
```

Quarantäne-Modus

Die Extraktoren funktionieren zwar relativ gut, dennoch werden öfters auch falsche Informationen identifiziert bzw. als Resultat zurückgegeben. Falls diese Daten direkt in die dafür vorgesehenen Modelle geschrieben würden, könnte bei einer neuen Iteration nur noch davon ausgegangen werden, dass diese Informationen korrekt sind. Mit einem Quarantäne-Modus soll verhindert werden, dass irrtümlich klassifizierte Daten sofort wiederverwendet werden. Hierzu wurde eine Konfigurationsmöglichkeit geschaffen, mit welcher neue Daten vorerst in eine separate Quarantäne-Datei abgelegt werden.

```
# machine learning mode
configuration.ml.quarantine=true
```

Dies führt dazu, dass neue Skills gemäss obiger Konfiguration z.B. «known_skills.txt.quarantine» abgelegt würden. Die gesammelten Resultate können nun manuell geprüft und in danach ergänz werden.

```
public static void addDataToTrainFile(String filename, String data) {

    try {

        // determine if quarantine is enabled
        boolean quarantine = ConfigurationUtil
                .get("configuration.ml.quarantine").equalsIgnoreCase("true");

        if (quarantine) {
            filename += ".quarantine";

            // make sure quarantine exists
            if (!Files.exists(Paths.get(filename))) {
                Files.createFile(Paths.get(filename));
            }
        }

        ...

        br.close();

        PrintWriter pw = new PrintWriter(filename);

        // persist content to file
        for (String entry : newContents) {
            pw.println(entry);
        }

        // print new data line
        pw.println(data);

        pw.flush();
        pw.close();

        LOG.debug("Saved found data in file " + filename);

    } catch (IOException e) {
        LOG.error("Something went wrong while learning new data.", e);
    }

}
```

5. Fazit & Ausblick

Das Ziel dieser Arbeit war es, eine Bibliothek zu entwickeln, welche im Stande ist, bestimmte Informationen aus Stelleninseraten zu extrahieren. Dieses Ziel wurde mit der abgelieferten Java-Bibliothek erreicht. Die Schwierigkeit liegt bei solchen Inseraten natürlich darin, eine möglichst hohe Trefferquote zu erreichen. Die entwickelte Bibliothek wies nach eigenen Messungen eine durchschnittliche Erfolgsquote von 85% auf. Aus unserer Sicht eine gute Quote, die das Erreichen der Zielsetzung bekräftigt.

Persönlich ziehen wir ein positives Fazit. Zu Beginn haben wir uns schwer damit getan, die Technologien im Bereich «Natural Language Processing» und «Machine Learning» zu verstehen. Nach Implementation von verschiedenen Prototypen konnten wir jedoch etwas Licht ins Dunkle bringen. Zu Beginn waren unsere Ansprüche, Informationen perfekt zu extrahieren, sehr hoch. Schnell mussten wir jedoch realisieren, dass in diesem Gebiet Perfektion nicht zu erreichen ist. Diese Erfahrungen sowie das neu dazu gelernte Wissen stellen für uns eine Bereicherung dar und wird in Zukunft sicher von Nutzen sein.

Folgende Verbesserungsvorschläge und Ideen könnten in einer weiteren Version sinnvoll sein.

Gebiete für Standortextraktion beachten

Kürzel von Kantonen oder Bundesländer könnten bei der Bestimmung bzw. Identifizierung eines Standortes einbezogen werden. Dies würde das Unterscheiden von Kandidaten verbessern.

Art der Anstellung beachten

Es kommt immer wieder vor, dass Stellen als «Festanstellung» oder «Temporär» bezeichnet werden. Eine Extraktion dieser Information wäre nützlich bei der gezielten Filterung von Stelleninseraten.

Teilzeit/Vollzeit beachten

Bei der Suche nach dem Pensum werden in der aktuellen Implementation nur numerische Werte beachtet. In Inseraten findet sich öfters auch eine Aussage, ob die Anstellung Voll- oder Teilzeit ist. Diese Information würde bei der Validierung der gefundenen Pensen helfen.

Indikatoren dynamischer machen

Manche Indikatoren wurden in der Implementation nicht dynamisch, sondern fest definiert. Durch eine Auslagerung dieser mittels maschinellem Lernen könnten Resultate weiter verbessert werden.

Weitere Eingabeformate

Die aktuelle Implementation lässt ausschliesslich die Analyse von Inseraten zu, welche durch einen Hyperlink definiert werden können und im HTML Format geliefert werden. Die Bibliothek wurde so entwickelt, dass auch andere Formate grundsätzlich zu positiven Ergebnissen führen können. So wäre eine Erweiterung nützlich, so dass z.B. auch Word- oder PDF-Dateien verarbeitet werden könnten.

Strategie zur strukturellen Analyse

Gesuchte Informationen häufen sich in Inseraten oft am selben oder an naheliegenden Positionen. Könnte eine Strategie entwickelt werden, welche ausgehend davon eher im Umfeld gefundener Informationen sucht, könnten störende Elemente besser ignoriert bzw. umgangen werden.

Erkennung von Skills verbessern

Die Erkennung der benötigten Fähigkeiten ist aktuell eher rudimentär implementiert. Skills welche bereits bekannt sind werden relativ gut erkannt. Bei der Suche nach neuen Skills wird jedoch davon ausgegangen, dass es sich meist um Nomen handelt. Kombinationen verschiedener Typen werden so nicht als zusammenhängender Skill erkannt. Dies führt dazu, dass entweder die Bedeutung verloren gehen kann oder die Fähigkeit als Ganzes erst gar nicht erkannt wird.

Job-Titel-Suffixe

Stellenbezeichnungen enden oft mit bestimmten Bezeichnungen, wie beispielsweise «Entwickler», «Manager» oder «Assistent». Um die Trefferquote der Extraktion von Stellenbezeichnungen zu erhöhen, kann eine Liste solcher Begriffe geführt und für die Extraktion verwendet werden.

Styling beachten

Visuell hervorgehobene Text-Elemente sind bei Stelleninseraten meistens von Bedeutung. Momentan wird dies berücksichtig, indem HTML-Elemente speziell behandelt werden, welche visuell hervorgehoben sind, wie beispielsweise Überschriften-Elemente. Wird aber ein «gewöhnliches» Text-Element durch CSS visuell hervorgehoben, wird das nicht erkannt. Um solche Fälle abzudecken, kann das CSS analysiert werden, um solche Elemente zu identifizieren.

Resultate untereinander vergleichen

Es gibt Fälle, in welchen die Resultate verbessert werden könnten, z.B. falls in einer Stellenbezeichnung der Name der Firma enthalten ist. Hier oder auch in weiteren Anwendungsfällen, würde es Sinn machen, die Resultate zu vergleichen. So könnten Überschneidungen erkannt und wenn nötig weitere Aktionen zur Verbesserung ausgeführt werden.

Kontext einer Information besser verstehen

Informationen werden oft falsch interpretiert. Beispielsweise werden Prozent-Werte oft als Pensum klassifiziert, obwohl sie eine andere Bedeutung hätten. Wenn der Kontext einer Information erkannt werden würde, könnte solche Fehlidentifizierungen verhindert werden.

6. Ehrlichkeitserklärung

Hiermit bestätigen die Autoren, diese Arbeit ohne fremde Hilfe und unter Einhaltung der gebotenen Regeln erstellt zu haben.

Kevin Kirn

Brugg, 21.12.2017

Ort, Datum | Unterschrift

Hoang Tran

Brugg, 21.12.2017

Ort, Datum | Unterschrift

7. Literaturverzeichnis

7.1. Internet

- Web Scraper Dokumentation (2017)
  http://webscraper.io/documentation (aufgerufen am 14.03.2017)

- Stanford JavaNLP API Documentation - CoreNLP (2017)
  https://nlp.stanford.edu/nlp/javadoc/javanlp/ (aufgerufen am 16.03.2017)

- Apache OpenNLP Developer Documentation (2017)
  https://-pennlp.apache.org/docs/1.8.3/manual (aufgerufen am 28.03.2017)

- LingPipe Dokumentation (2017)
  http://alias-i.com/lingpipe/demos/tutorial/read-me.html (aufgerufen am 18.04.2017)

- Spacy.io API Reference (2017)
  https://spacy.io/api (aufgerufen am 26.05.2017)

- GeoAdmin API Dokumentation (2017)
  https://api3.geo.admin.ch/api/doc.html (aufgerufen am 03.08.2017)

7.2. Bücher

- Dengel, Andreas (2011): Semantische Technologien. 2., neu bearbeitete und aktualisierte Auflage. Heidelberg: Spektrum Akademischer Verlag

- Ethem Alpaydin (2008): Maschinelles Lernen. 2., neu bearbeitete und aktualisierte Auflage. München: Oldenbourg

8. Abbildungsverzeichnis

9. Glossar

Begriff	**Definition**
Dictionary	Liste von Objekten, welche für die Suche optimiert ist. Wird im Rahmen des Projekts in Zusammenhang für den Vergleich eines Begriffs mit bekannten Begriffen verwendet.
Digitale Revolution	Bezeichnet den durch die Digitalisierung und Computern ausgelösten Umbruch in der Gesellschaft.
DOM	Abkürzung für «Document Object Model». Ein DOM ist die Repräsentation eines HTML-Dokuments als Baumstruktur. Jeder Knoten im Baum ist ein Objekt des Dokuments, wie beispielsweise eine Überschrift oder ein Absatz.
Fuzzy Search	Ungenaue Suche nach Zeichenketten welche ähnlich sind.
HashMap	Eine HashMap ist im Prinzip eine Liste von Schlüssel-Wert-Paaren. Jeder Eintrag im HashMap besitzt somit einen Schlüssel und einen Wert. Jeder Schlüssel ist hierbei eindeutig, sodass niemals mehrere Einträge die gleichen Schüsseln haben können. Es können aber durchaus mehrere Einträge den gleichen Wert haben, solange die Schlüssel sich unterscheiden.
NER	Abkürzung für «Named Entity Recognition». Aufgabengebiet im Bereich der Computerlinguistik, mit welcher Entitäten anhand von Modellen in einem Text identifiziert werden können.
OCR	Abkürzung für «Optical Character Recognition», zu Deutsch «Optische Zeichenerkennung». Das ist ein Begriff aus der Informationstechnik und bezeichnet die durch Maschinen automatisierte Texterkennung innerhalb von Bildern.
Pattern Matching	Textsuche nach einem Muster, also keinem direkten Wort sondern deren definierten Struktur. Beispiel PLZ Format «XXXX Ortsname».
PoS	Abkürzung für «Part of Speech» (Analyse). Grammatikalische Analyse von Wortgruppen und den gemeinsamen Abhängigkeiten.
Recruiting	Personenbeschaffung, Anwerbung von Mitarbeitern
Regex	Regex ist die Kurzform von «Regular Expression» und ist die englische Bezeichnung von «Regulärer Ausdruck». Siehe «Regulärer Ausdruck»
Regulärer Ausdruck	Ein Muster, welche bestimmten Worte oder Wortgruppen beschreibt und zur Suche nach diesen verwendet werden kann.