Bachelorarbeit

Erweitern von Aqqu durch AutoSklearn

Daniel Bindemann

1. Gutachter: Prof. Dr. Hannah Bast
2. Gutachter: Prof. Dr. Frank Hutter

Albert-Ludwigs-Universität Freiburg
Technische Fakultät
Institut für Informatik
Lehrstuhl für Algorithmen und Datenstrukturen

23. 12. 2016

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Ort, Datum	Unterschrift

Inhaltsverzeichnis

Tabellenverzeichnis

1 Einleitung

Seien es autonom fahrende Autos, Systeme für präzise Gesichtserkennung oder intelligente Anti-Spam-Filter – Eine Vielzahl heutiger Technologien verdankt ihren Erfolg den Fortschritten im Bereich des maschinelles Lernens. Ein Gebiet der Informatik, dessen Erfolg unter anderem auch auf maschinellem Lernen basiert, ist das *Natural Language Processing* (zu Deutsch auch als *Computerlinguistik* bezeichnet). In diesem Gebiet wird versucht, natürlichsprachliche Ausdrücke mit Hilfe des Computers zu analysieren und deren Bedeutung zu interpretieren, um zum Beispiel auf natürlichsprachliche Fragen die passende Antwort in einer Sammlung von Daten finden zu können. Insbesondere für Nutzer, welche nicht im Bereich der Informatik versiert sind und ihre Fragen nicht in formalen Abfragesprachen wie SQL oder SPARQL ausdrücken können, könnten solche Systeme es möglich machen, benötigte Informationen schnell und einfach durch einen kurzen Satz im Internet ausfindig zu machen. Aber auch Nutzer, welche formale Abfragesprachen beherrschen, könnten von solchen Systemen profitieren, da sie ohne Wissen über die genaue Art der Datenspeicherung und ohne umständliche Anfragen effizient Informationen aus Wissensdatenbanken extrahieren könnten.
Da das maschinelle Lernen ein zentraler Bestandteil für den Erfolg solcher natürlichsprachlicher Suchmaschinen ist und in vielen kürzlich entworfenen Systemen verwendet wird, wäre es ein sinnvoller Versuch, diese Systeme durch Fortschritte im Bereich des maschinellen Lernens zu erweitern.
Das Ziel dieser Arbeit ist es, ein erfolgreiches System für das Beantworten natürlichsprachlicher Fragen, *Aqqu*, mit einem modernen Toolkit für automatisiertes maschinelles Lernen, *auto-sklearn*, zu erweitern. Dadurch soll erreicht werden, dass Aqqu einen größeren Anteil an Fragen korrekt beantworten kann bzw. bei Fragen mit mehreren möglichen Antworten die Menge der korrekten Antworten präziser ausfindig machen kann.
Im Folgenden stelle ich zunächst die Funktionsweise und wichtigsten Features dieser zwei Systeme vor (zusammengefasst aus den zugehörigen Papers [1] und [2]). Im Anschluss beschreibe ich meine Ansätze, um die Systeme miteinander zu verbinden und evaluiere schließlich die Ergebnisse dieser Ansätze.

2 Vorstellung von Aqqu und auto-sklearn

2.1 Aqqu

Aqqu beantwortet gegebene Fragen, indem es diese in eine passende SPARQL-Anfrage übersetzt, und damit aus einer Wissensdatenbank (hier wird *Freebase* verwendet) die gesuchte(n) Antwort(en) extrahiert. Der Prozess, aus natürlichsprachlichen Fragen eine passende SPARQL-Anfrage zu erzeugen, lässt sich grob in folgende Schritte aufteilen:

1. **Identifizierung der Entitäten**
   Zunächst wird versucht, die Entitäten aus der Wissensdatenbank zu finden, welche in der Frage erwähnt werden. Aufgrund der Existenz mehrdeutiger Wörter und Synonyme ist es nicht immer möglich, die korrekten Entitäten eindeutig zu bestimmen. Es wird daher in diesem Schritt eine ganze Liste von Entitäten erzeugt, und jeder Entität eine Punktzahl zugewiesen. Die Punktzahl ist dann ein Maß für die Wahrscheinlichkeit, dass diese Entität in der Frage erwähnt wird.

2. **Generierung von Kandidaten**
   Für das Beantworten der Frage wird als nächstes eine Liste von Kandidaten erzeugt, wobei jeder Kandidat einer SPARQL-Anfrage entspricht, mit der eine Antwort aus der Wissensdatenbank extrahiert werden kann. Dafür gibt es drei mögliche Schemata, welche jeweils eine Vorlage für einen Teilgraph der Wissensdatenbank darstellen und aus unterschiedlicher Anzahl von Entitäten und Relationen bestehen. Durch passendes Einsetzen der im ersten Schritt gefundenen Entitäten in diese Schemata erhält man eine Menge von Kandidaten.

3. **Bewerten der Relationen**
   In diesem Schritt werden die Relationen bewertet, welche die Entitäten in den Kandidaten miteinander verbinden (bei mehr als 2 Entitäten über einen Mediator). Dafür werden die in der Frage vorkommenden Wörter nach mehreren unterschiedlichen Kriterien mit den Namen der Relationen in den Kandidaten verglichen:

a) Wörtliche Übereinstimmung: Stimmen beide Wörter exakt überein?

b) Synomymität: Haben beide Wörter die selbe oder sehr ähnliche Bedeutung?

c) Ableitung: Lässt sich ein Wort aus dem anderen ableiten (z.B. Substantivierung des anderen Wortes)

d) Ähnlicher Kontext: Tauchen beide Wörter öfters im selben Kontext auf und sind sie auch in der Wissensdatenbank direkt oder indirekt miteinander eng verbunden?

4. **Finden von N-Gramm-Korrespondenzen**
   In diesem Schritt werden Korrespondenzen zwischen einzelnen Wörtern (Monogrammen) oder Kombinationen aus zwei Wörtern (Bigrammen) aus der Frage und den Relationen in den Kandidaten gesucht (z.B. könnte ein Bigramm 'Wer ist' auf eine Relation 'Profession' hindeuten – Solche Korrespondenzen werden nicht durch die vier Kriterien im vorherigen Schritt entdeckt). Hierfür wird jede mögliche N-Gramm-Korrespondenz durch ein Merkmal dargestellt, welches aus dem Monogramm oder Bigramm in der Frage und den Relationsnamen im Kandidaten besteht. Für jeden Kandidaten erhält man somit einen sehr dünnbesetzten Merkmalsvektor, in welchem fast alle Einträge 0 sind, und eine 1 nur bei den Merkmalen steht, dessen Kombination aus N-Gramm und Relationsnamen für diesen Kandidaten vorkommt.
   Ein Klassifikator wird auf der Menge dieser Merkmalsvektoren trainiert, wobei die Vektoren korrekter Kandidaten als positive Trainingsbeispiele dienen, und der Rest als negative. Dieser Klassifikator weist dann jedem Kandidaten eine Wahrscheinlichkeit zu, welche darstellt, wie stark die N-Gramm-Korrespondenzen zwischen diesem Kandidaten und der Frage sind. Um Overfitting zu vermeiden, wird der Trainings-Datensatz in sechs Teile gespalten, und dann für jeden Teil die N-Gramm-Korrespondenzen berechnet, nachdem auf den übrigen 5 Teilen trainiert wurde.

5. **Erzeugen von Merkmalsvektoren**
   Für jeden Kandidaten können die in den vorherigen Schritten gesammelten Merkmale (Bewertung der Entitäten, Bewertungen der Relationen, N-Gramm-Korrespondenzen) zu einem Merkmalsvektor vereinigt werden. Zu diesem werden noch weitere Merkmale hinzugefügt, z.B.:

   - Kombinationen von bisher gesammelten Merkmalen
   - Anzahl der Ergebnisse, die ein Kandidat beim Ausführen der zugehörigen SPARQL-Anfrage aus der Wissensdatenbank extrahiert

6. **Pruning der Kandidaten**
   Bisher wurden für jeden Kandidaten nur eine Reihe von Merkmalen gesammelt, aber noch keine Gesamtbewertung nach Relevanz durchgeführt. In diesem Schritt werden Kandidaten, welche mit hoher Wahrscheinlichkeit nicht relevant sind (d.h. keine passende Antwort auf die Frage geben), direkt verworfen. Dadurch wird es möglich, auf eine Frage auch keine Antwort geben zu können, wenn alle Kandidaten irrelevant sind. Dies ist insbesondere nützlich für den Fall, dass keine passende Antwort auf eine Frage in der Wissensdatenbank vorhanden ist.
   Für diesen Zweck wird ein Pruning-Klassifikator auf den Merkmalsvektoren der Kandidaten trainiert: Die Kandidaten mit der korrekten Antwort auf die zugehörige Frage zählen dabei als positive Beispiele, Kandidaten mit der falschen Antwort als negative. Kandidaten, welche der Pruning-Klassifikator als negativ klassifiziert, werden dann als irrelevant betrachtet und verworfen. Damit nur die schlechtesten Kandidaten als irrelevant klassifiziert werden und keine wichtigen Kandidaten verworfen werden, werden beim Training die korrekten Kandidaten doppelt so stark gewichtet wie die negativen Kandidaten.

7. **Ranking der Kandidaten**
   Nach dem Pruning werden die übrigen Kandidaten schließlich nach Relevanz geordnet. Der Kandidat, welcher in dieser Ordnung ganz oben steht, wird dann ausgewählt, um eine Antwort auf die Frage zu liefern.

   Das Ordnen der Kandidaten wird wieder durch einen Klassifikator gelernt. Es werden zwei verschiedene Ansätze für das Ranking dieser Kandidaten betrachtet:

   - **Punktweises Ranking:**
     Bei diesem Ansatz wird für jeden Kandidaten separat eine Wahrscheinlichkeit dafür berechnet, dass dieser das korrekte Ergebnis liefert. Nach diesem Wahrscheinlichkeitswert können die Kandidaten dann sortiert werden.
     Das Training auf den Merkmalsvektoren funktioniert wie beim Pruning-Klassifikator (welcher in Schritt 6 beschrieben wurde), hier aber mit gleicher Gewichtung für alle Kandidaten.
   - **Paarweises Ranking:**
     Statt einer Wahrscheinlichkeit für jeden einzelnen Kandidaten wird hier eine Vergleichsfunktion gelernt, welche ein Paar von Kandidaten als Eingabe nimmt und entscheidet, welcher von diesen zwei Kandidaten eine bessere Antwort liefern kann. Durch einen vergleichsbasierten Sortieralgorithmus kann somit eine Ordnung auf der Menge der Kandidaten her-

gestellt werden.
Daten für das Training können konstruiert werden, indem man die Differenz des Merkmalsvektors vom korrekten Kandidaten einer Frage mit dem Merkmalsvektor eines anderen Kandidaten zur selben Frage berechnet. Ein positives Trainingsbeispiel ist dann ein Tupel bestehend aus diesem Differenzvektor und den individuellen Merkmalsvektoren der zwei gewählten Kandidaten. Für ein negatives Trainingsbeispiel verwendet man den Gegenvektor des Differenzvektors.
Auf diese Art und Weise lässt sich für jeden inkorrekten Kandidaten sowohl ein positiver als auch negativer Datenpunkt für das Training konstruieren. Um genügend Datenpunkte zu erhalten, werden die Hälfte der inkorrekten Kandidaten für die Konstruktion verwendet, mindestens aber 200 (oder alle, falls weniger als 200 vorhanden sind).

2.2 auto-sklearn

auto-sklearn ist ein System für automatisiertes maschinelles Lernen. Es stellt einen Klassifikator bereit, welcher versucht, auf einem gegebenen Datensatz automatisch eine Konfiguration zu finden, mit welcher das dem Datensatz unterliegende Modell am besten erlernt werden kann.
Eine solche Konfiguration beschreibt eine Pipeline, welche aus mehreren Teilen besteht:

- Bis zu drei Daten-Präprozessoren (von insgesamt 4 verfügbaren). Diese Präprozessoren transformieren die Werte der Merkmale in gegebenen Daten überall, wo sie angewendet werden können. Z.B. können fehlende Werte in den Daten mit künstlich generierten Werten gefüllt werden (Imputation) oder skaliert werden, um in eine passende Form für den Klassifikator gebracht zu werden.

- Ein optionaler Merkmals-Präprozessor (von insgesamt 14 verfügbaren). Diese Präprozessoren verändern nicht nur die Werte von Merkmalen, sondern auch die Menge der Merkmale selbst. Z.B. können die Daten in eine Form projiziert werden, in welcher sie linear trennbar sind.

- Ein Klassifikator (von insgesamt 15 verfügbaren). Dieser verwendet die in den vorherigen Schritten transformierten Daten, um schließlich das Modell zu erlernen. Beispiele dafür sind Random Forests, SVMs (Support Vector Machines) und Gradient Boosting.

In den obigen Schritten müssen passende Präprozessoren bzw. Klassifikatoren gewählt werden, und jedes dieser Objekte hat wiederum eine eigene Menge an Parametern,

die auch passend gewählt werden müssen. Dadurch ergibt sich ein mehrdimensionales Optimierungsproblem, bestehend aus 110 Hyperparametern, welches in auto-sklearn iterativ durch *Bayesian Optimization* gelöst wird. Dieses Optimierungsverfahren wird in auto-sklearn mit zwei weiteren Methoden erweitert:

- **meta learning**: Durch Anwendung von *meta learning* werden potenziell gute Start-Konfigurationen für das Optimierungsverfahren gesucht. Dazu wurden im Voraus verschiedene Konfigurationen auf bereits bekannten Datensätzen evaluiert. Betrachtet man nun einen neuen Datensatz, kann man ihn anhand seiner Eigenschaften (*meta-features*) mit den bekannten Datensätzen vergleichen und daraus Schlüsse ziehen, welche Konfigurationen auch für den neuen Datensatz relativ gute Ergebnisse liefern könnten.

- **ensemble selection**: Um robustere Modelle zu erhalten, wird beim Optimierungsverfahren nicht nur das Modell mit dem besten Ergebnis gewählt, sondern eine Kombination aus mehreren unterschiedlich gewichteten Modellen, welche vergleichsweise gut abgeschnitten haben. Für das Erstellen dieses Ensembles wird der *ensemble selection* Algorithmus verwendet, in welchem iterativ die Modelle zum Ensemble hinzugefügt werden, welche die Leistung des Ensembles in der jeweiligen Iteration maximieren. Dabei können Modelle auch wiederholt hinzugefügt werden, wodurch die Gewichtung dieses Modells im Ensemble erhöht wird.

3 Ansatz

An der Beschreibung von Aqqu in Abschnitt 2.1 kann man erkennen, dass maschinelles Lernen an mehreren Stellen im System verwendet wird:

- Lernen eines Klassifikators, welcher jedem Kandidaten einen Wert für die N-Gramm-Korrespondenzen zuweist.
- Lernen eines Pruning-Klassifikators, welcher Kandidaten mit sehr niedriger Punktzahl aussondert.
- Ranking der Kandidaten nach Relevanz, um den Kandidaten zu finden, der die Frage am besten beantworten kann.

Der zentrale und wichtigste Schritt ist dabei das Ranking, für welches ich im Folgenden meine Ansätze beschreiben werde.

3.1 Ranking

Aqqu benutzt standardmäßig als Klassifikator einen Random Forest für das Ranking der Kandidaten. Ich ersetze diesen durch einen Klassifikator von auto-sklearn unter Verwendung der Standardoptionen.
Zusätzlich passe ich auch die Metrik an, nach welcher Konfigurationen in auto-sklearn bei dem Training evaluiert werden. Standardmäßig wird als Metrik *accuracy* verwendet, d.h. die Leistung einer Konfiguration von Hyperparametern bzw. eines Ensembles von Modellen wird nach dem Anteil richtig klassifizierter Datenpunkte bewertet (Für die Evaluation in Abschnitt 4 bezeichne ich diese Metrik als *Standard-Accuracy*). In Aqqu stellen diese Datenpunkte Kandidaten für die Beantwortung von Fragen dar, bzw. Vergleiche zwischen Kandidaten im Falle von paarweisem Ranking, wie in Abschnitt 2.1 unter dem Punkt *Ranking der Kandidaten* beschrieben. Diese Kandidaten können für unterschiedliche Datenpunkte jedoch zu unterschiedlichen Fragen gehören – Eine maximale Gesamtzahl an richtig klassifizierten Datenpunkten stimmt daher nicht unbedingt mit einer maximalen Anzahl richtig beantworteter Fragen überein.

Aus diesem Grund passe ich auto-sklearn so an, dass nach dem maximalen Anteil an richtig beantworteten Fragen optimiert wird, statt nach korrekter Klassifikation

der einzelnen Kandidaten zu verschiedenen Fragen. Zunächst speichere ich für jeden Datenpunkt aus dem Datensatz zusätzlich die Information, zu welcher Frage der Kandidat bzw. das Paar von Kandidaten gehört, welches von diesem Datenpunkt repräsentiert wird. Im Falle von Punktweisem Ranking lässt sich bei der Berechnung der Metrik mit dieser Information für jede Frage der Kandidat bestimmen, welcher vom betrachteten Modell für die Antwort auf die Frage gewählt werden würde (nämlich der Kandidat gehörend zum Datenpunkt, der unter allen anderen Datenpunkten zur selben Frage mit der höchsten Wahrscheinlichkeit als positiv klassifiziert wurde). Ist dieser Kandidat der Kandidat mit der korrekten Antwort auf die Frage, zählt die Frage als korrekt beantwortet. Die Metrik ergibt sich dann als der Anteil der Fragen, welche auf diese Weise korrekt beantwortet wurden. Für die Evaluation bezeichne ich diese Metrik mit *Question-Accuracy-Exact*.

Bei paarweisem Ranking ist es aber nicht praktikabel, bei jeder Evaluation eines Ensembles oder einer Konfiguration alle Fragen erneut komplett zu evaluieren, da dies viel Zeit in Anspruch nehmen würde. Statt der eigentlichen Metrik verwende ich aus diesem Grund nur eine Heuristik, welche sich komplett mit gegebenen Daten (Teil vom Trainings-Datensatz bestehend aus paarweisen Vergleichen von Kandidaten, wie in Abschnitt 2.1 beschrieben) berechnen lassen kann: Für jede Frage bestimme ich, wie viel Prozent der Datenpunkte für diese Frage korrekt klassifiziert wurde, d.h. bei wie vielen Vergleichen zwischen Paaren von Kandidaten korrekt vorhergesagt wurde, welcher Kandidat der Bessere ist. Eine Frage zähle ich bei dieser Heuristik dann als korrekt beantwortet, falls alle Datenpunkte zu dieser Frage korrekt klassifiziert wurden. Die Metrik, welche ich mit *Question-Accuracy* bezeichne, ergibt sich somit als der Anteil der Fragen, welche auf obige Weise korrekt beantwortet wurden.

Zusätzlich implementiere ich eine weitere Metrik mit dem Namen *Question-F1-Exact*. Es gibt nämlich einen Datensatz von Fragen und Antworten, bei welchem das Ergebnis üblicherweise nicht nach Accuracy, sondern nach mittlerem F1-Wert bewertet wird (siehe Abschnitt 4 für genauere Informationen zu diesem Datensatz). Die Metrik *Question-F1-Exact* definiere ich ähnlich zur Metrik *Question-Accuracy-Exact* (Und lässt sich auch nur für punktweises Ranking definieren): Für jede Frage suche ich wieder den Kandidaten, welcher mit der höchsten Wahrscheinlichkeit als positiv klassifiziert wurde, und berechne dann auf der Antwortmenge dieses Kandidaten den F1-Wert. Der Durschnitt dieser F1-Werte für alle Fragen bildet dann den Wert, den diese Metrik liefert. Dadurch ist es möglich, nicht nach Accuracy, sondern nach mittlerem F1-Wert passend zum Datensatz zu optimieren.

4 Evaluation

Um die Leistung von Aqqu zu beurteilen, werden Datensätze mit Fragen und deren zugehörigen Antworten benötigt. Aqqu verwendet dafür die Datensätze *Free917* und *WebQuestions*, welche ich auch hier für die Evaluation meiner Anpassungen verwenden werde.
Free917 besteht aus 917 Fragen mit jeweils einer SPARQL-Anfrage, mit welcher man die korrekten Antworten aus der Freebase-Wissensdatenbank erhalten kann. Da diese Fragen spezifisch aus den Daten von Freebase nach einem bestimmten Schema (nämlich immer genau eine Relation, möglicherweise n-teilig über eine Mediator-Entität) erzeugt wurden, lässt sich jede Frage exakt durch eine Menge von Entitäten aus Freebase beantworten. Als Metrik für die Auswertung wird daher üblicherweise *Accuracy* gewählt, d.h. der Anteil exakt korrekt beantworteter Fragen.
WebQuestions besteht aus 5,810 Fragen mit jeweils einer zugehörigen Antwortmenge. Diese Fragen wurden nicht einzeln von Hand erstellt, sondern aus Suchvorschlägen von Google generiert und per Crowdsourcing mit Antworten ergänzt. Aus diesem Grund sind nicht immer alle Antworten in Freebase enthalten und auch fehlerhafte Antworten unter den korrekten. Da sich viele Fragen somit nicht exakt beantworten lassen, wird als Metrik hier üblicherweise der *mittlere F1-Wert* gewählt, wodurch auch teilweise beantwortete Fragen in das Ergebnis miteinbezogen werden können. Für jede Frage wird dafür der F1-Wert für die gegebene Antwortmenge berechnet, und der Durchschnitt dieser F1-Werte für alle Fragen dann als Metrik für die Evaluation verwendet.
Aufgrund begrenzter Rechenressourcen evaluiere ich hier nur auf einem zufällig gewählten Teil des gesamten WebQuestions-Datensatzes, bestehend aus 755 Fragen für das Training und 304 Fragen für die Evaluation.

Die Ergebnisse für die Anpassungen des Rankings der Kandidaten, beschrieben in Kapitel 3.1, werden für den Datensatz Free917 in Tabelle 4.1 dargestellt, und für den Datensatz WebQuestions in Tabelle 4.2. Die Metrik, nach welcher in auto-sklearn optimiert wurde, steht dabei in Klammern (mit der selben Terminologie wie in Kapitel 3.1).
Man beachte, dass die Ergebnisse aufgrund von Zufallsfaktoren beim Training mit auto-sklearn schwanken. Um auto-sklearn genügend Zeit für das Training zu geben

	Punktweise	Paarweise
Aqqu	59,058	65,217
Aqqu+AutoSklearn (Standard-Accuracy)	66,304	66,243
Aqqu+AutoSklearn (Question-Accuracy)	62,319	66,6148
Aqqu+AutoSklearn (Question-Accuracy-Exact)	61,051	-

Tabelle 4.1: Ergebnisse auf dem Test-Datensatz von Free917 (267 Fragen). Die Werte stehen für *accuracy* auf Fragen in Prozent.

	Punktweise	Paarweise
Aqqu	42,461	46,171
Aqqu+AutoSklearn (Standard-Accuracy)	43,823	46,927
Aqqu+AutoSklearn (Question-Accuracy)	43,17	46,716
Aqqu+AutoSklearn (Question-F1-Exact)	40,3852	-

Tabelle 4.2: Ergebnisse auf dem Test-Datensatz von einem Teil von WebQuestions (304 Fragen). Die Werte stehen für den *mittleren F1-Wert* auf Fragen in Prozent.

(standardmäßig eine Stunde) und genügend Testläufe bei der Evaluation durchführen zu können (Durchschnitt über 10 bis 20 Testläufe), wird für jede Komplettevaluation (Training und Evaluation) bis zu zwei Stunden benötigt. Aufgrund der langen Evaluationszeit habe ich für jede Konfiguration daher nur wenige Komplettevaluationen durchgeführt. Die Werte in der Tabelle sind somit Durchschnittswerte über 3–7 Werte, weshalb man kleine Abweichungen durch Zufall nicht ausschließen kann.

Man erkennt schnell, dass paarweises Ranking in fast jedem Fall bessere Ergebnisse liefert als punktweises Ranking, insbesondere auf WebQuestions (Nur bei der Version mit auto-sklearn und der Standard-Accuracy auf Free917 sind die Ergebnisse fast die selben). Dies stimmt überein mit den Ergebnissen in [1] und der Intuition, dass paarweises Ranking besser ist. Im jedem Vergleich von zwei Kandidaten werden nämlich nur Kandidaten zu der selben Frage miteinander verglichen, während bei punktweisem Ranking auf den Kandidaten von allen Fragen zusammen trainiert wird, obwohl die Fragen sehr unterschiedlich sein können und nicht unbedingt miteinander verglichen werden sollten.

Man erkennt an den Ergebnissen auch, dass auto-sklearn wie erwartet mit der Standard-Metrik im Vergleich zum ursprünglichen Aqqu das Ergebnis immer verbessert. Die Ergebnisse des punktweisen Rankings bekommen auf dem Datensatz Free917 mit auto-sklearn einen Anstieg von ganzen sieben Prozent, auf WebQuesti-

ons sind es rund 1,4 Prozent. Im Falle von paarweisem Ranking wird das Ergebnis auch verbessert, hier aber nur um 0,8 bis 1 Prozent.

Überraschend sind die Ergebnisse mit den angepassten Metriken, welche nicht auf Datenpunkten, sondern auf Fragen optimieren. Bei dem punktweisen Ranking auf Free917 sinken dort die Ergebniswerte um mehrere Prozent, und die approximierte Metrik (Question-Accuracy) ergibt bessere Ergebnisse als die exakte Metrik; Auf WebQuestions sieht es mit punktweisem Ranking ähnlich aus, obwohl die approximierte Metrik dort noch recht gute Ergebnisse liefert.

Bei paarweisem Ranking ergeben die angepassten Metriken fast die selben Ergebnisse wie die Standard-Metrik. Da die Werte bei der Evaluation schwanken, lässt sich ein exaktes Ergebnis zwar nicht feststellen, aber bei Free917 scheint die Question-Accuracy das Ergebnis etwas zu verbessern, während es bei WebQuestions ein klein wenig verschlechtert wird.

5 Zusammenfassung

Ich habe ein System für das Beantworten natürlichsprachlicher Fragen, *Aqqu*, auf zwei verschiedenen Sammlungen von Fragen und Antworten, *Free917* und *WebQuestions*, evaluiert. Mit dem Ziel, die Beantwortung der Fragen in Aqqu zu verbessern, habe ich ein System für automatisiertes maschinelles Lernen, *auto-sklearn*, in Aqqu integriert und verschiedene Varianten damit implementiert.

In Aqqu werden in einer Reihe von Schritten Antwort-Kandidaten generiert, welche aus der Wissensdatenbank *Freebase* eine Antwort auf eine gegebene Frage extrahieren können. In mehreren Schritten wird dabei maschinelles Lernen verwendet, um die Relevanz dieser Kandidaten zu bewerten. Insbesondere dem letzten Schritt, dem Ranking der Kandidaten, kommt dabei eine große Bedeutung zu und dessen Verbesserung mit Hilfe von auto-sklearn war auch der wesentliche Teil dieser Arbeit. Ich habe dafür einige Metriken implementiert, nach welchen das Ranking in auto-sklearn optimiert werden sollte. Diese Metriken habe ich auf zwei Ansätze für das Ranking, nämlich punktweises und paarweises Ranking, angewendet, und die Ergebnisse davon evaluiert.

Dabei hat sich gezeigt, dass unter Verwendung von auto-sklearn mit der Standardmetrik (*accuracy*) die Ergebnisse in allen Fällen sichtbar verbessert werden können: Im Falle von paarweisem Ranking um rund ein Prozent, im Falle von punktweisem Ranking um über ein Prozent auf WebQuestions und ganze sieben Prozent auf Free917.
Die neu implementierten Metriken, welche nach Accuracy oder mittlerem F1-Wert auf ganzen Fragen statt einzelnen Datenpunkten optimieren, haben im Fall von punktweisem Ranking schlechter abgeschnitten (teils um mehrere Prozent). Bei dem paarweisen Ranking konnte man durch diese Metriken auch nur sehr kleine Verbesserungen von einem halben Prozent auf Free917 erhalten, während die Ergebnisse auf WebQuestions fast die selben sind.

Literaturverzeichnis

[1] Bast, H., Haussmann, E.: More accurate question answering on freebase. In: CIKM '15 Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM (2015) 1431–1440

[2] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., eds.: Advances in Neural Information Processing Systems 28. Curran Associates, Inc. (2015) 2962–2970