Pipelines für effiziente und robuste Ad-hoc-Textanalyse[1]

Henning Wachsmuth[2]

Abstract: Suchmaschinen und Big-Data-Analytics-Anwendungen zielen darauf ab, ad-hoc relevante Informationen zu Anfragen zu finden. Häufig müssen dafür große Mengen natürlichsprachiger Texte verarbeitet werden. Um nicht nur potentiell relevante Texte, sondern direkt relevante Informationen zu ermitteln, werden Texte zunehmend tiefer analysiert. Dafür können theoretisch komplexe Pipelines mit zahlreichen Analysealgorithmen eingesetzt werden. Aufgrund fehlender Effizienz und Robustheit sind die durchgeführten Textanalysen in der Praxis jedoch bislang auf einfache, manuell erstellte Pipelines für antizipierte Anfragen beschränkt. Der vorliegende Beitrag gibt einen Überblick über einen Ansatz zur automatischen Erstellung von Pipelines für beliebige Textanalysen. Die resultierenden Pipelines sind effizienzoptimiert und arbeiten robust auf heterogenen Texten. Der Ansatz kombiniert zu diesem Zweck neuartige Verfahren, die auf Techniken der klassischen künstlichen Intelligenz und des maschinellen Lernens basieren. Formale Untersuchungen wie auch zahlreiche empirische Experimente belegen, dass der Ansatz einen wichtigen Schritt hin zum Ad-hoc-Einsatz von Textanalyse-Pipelines in Suchmaschinen und Big-Data-Analytics-Anwendungen darstellt.

Keywords: Textanalyse, Pipeline, Effizienz, Künstliche Intelligenz, Maschinelles Lernen

1 Einführung

Informationssuche ist ein fester Bestandteil des Lebens. Menschen surfen im Netz, um neues Wissen zu erlangen, und Unternehmen analysieren Big Data, um geschäftsrelevante Erkenntnisse zu gewinnen. Bereits heute finden Suchmaschinen in vielen Fällen die relevantesten Ergebnisse *ad-hoc*, das heißt, unmittelbar als Antwort auf eine Suchanfrage. Anstatt Informationen zurückzugeben, liefern sie jedoch meist nur zahlreiche Links zu Webseiten. Dies erschwert die Suche vor allem, wenn viele Informationen zusammenzubringen sind oder wenn sie einer Nadel im Heuhaufen gleicht. Big-Data-Analytics-Anwendungen stehen vor ähnlichen Problemen; geschätzte 95% aller geschäftsrelevanten Informationen liegen in Textform vor [HP10]. Daher sind Experten der Meinung, dass zukünftig mehr und mehr direkt relevante Informationen aus großen Mengen von Texten entnommen und aufbereitet werden [Et11, KH13]. Hierfür sind verschiedene Algorithmen zur Verarbeitung natürlicher Sprache nötig, hier unter dem Begriff *Textanalysen* zusammengefasst.

Die Ermittlung relevanter Informationen erfordert in der Regel eine Reihe aufeinander aufbauender Algorithmen. Um zum Beispiel Informationen über die Entwicklung einer Branche zu erhalten, müssen Relationen zwischen verschiedenen Typen von Entitäten (z.B. Firmennamen) und Ereignissen (z.B. Umsatzprognosen) erkannt und später normalisiert und aggregiert werden. Die Erkennung wiederum erwartet meist eine syntaktische Aufbereitung der Texte, etwa um Wortarten oder Satzglieder. Dafür müssen die Texte zuvor in Wörter und Sätze zerlegt worden sein. Aufgrund solcher Abhängigkeiten werden die

[1] Englischer Titel der Dissertation: "Text Analysis Pipelines—Towards Ad-hoc Large-scale Text Mining"

[2] Bauhaus-Universität Weimar, F. Medien, Bauhausstr. 11, 99423 Weimar, henning.wachsmuth@uni-weimar.de

verwendeten Algorithmen typischerweise in einer *Pipeline* angeordnet, die jeden Algorithmus nacheinander auf einem Text ausführt. Während für viele Textanalysen effektive Algorithmen existieren, werden Textanalyse-Pipelines in der Praxis bislang nur für einfache und antizipierte Suchanfragen eingesetzt. Dies hat insbesondere drei Gründe:

Erstens werden Pipelines bislang manuell (möglicherweise werkzeugunterstützt [Ka10]) für gegebene Anfragen erstellt, weil ihr Design Expertenwissen über die zu verwendenden Algorithmen erfordert. Daher können unvorhergesehene Anfragen nicht ad-hoc beantwortet werden. Zweitens mangelt es vielen Pipelines an Laufzeiteffizienz, weil ihre Ausführung die Analyse von Texten mit berechnungsintensiven Algorithmen beinhaltet [Sa08]. Daher eignen sie sich nicht für den Einsatz auf großen Datenmengen. Und drittens sind viele Pipelines nicht robust gegenüber der Vielfalt natürlicher Sprache, weil die verwendeten Algorithmen domänenspezifische Textmerkmale analysieren [BDP07]. Daher lässt sich im Fall beliebiger Texte keine hinreichende Effektivität gewährleisten.

In der Dissertation, die diesem Beitrag zugrunde liegt, wird die Fragestellung untersucht, wie sich Textanalysen ad-hoc in effizienter und robuster Weise durchführen lassen [Wa15]. Zentrale Hypothese ist, dass sich Wissen über eine Textanalyse sowie Informationen, die während der Textanalyse gewonnen werden, nutzen lassen, um das Design, die Ausführung und die Ausgabe der eingesetzten Pipeline zu verbessern. Um dies automatisch zu erreichen, werden Techniken der klassischen künstlichen Intelligenz und des maschinellen Lernens eingesetzt. Der in der Dissertation vorgestellte vierteilige Ansatz und die erreichten Ergebnisse werden auf den folgenden Seiten zusammengefasst.

Im Speziellen dient der Stand der Technik von Pipelines innerhalb der gegebenen Problemstellung als Ausgangspunkt (Abschnitt 2). Zuerst werden wissensbasierte Verfahren zur Ad-hoc-Erstellung einer Pipeline für eine gegebene Anfrage sowie zur optimalen Ausführung der Pipeline auf ihren Eingabetexten erarbeitet. Darauf aufbauend wird gezeigt, wie sich die Reihenfolge der verwendeten Algorithmen sowohl theoretisch als auch praktisch auf Basis der Informationen in Texten optimieren lässt, um die Laufzeiteffizienz der Pipeline zu maximieren. Schließlich wird eine neuartige Mustererkennung entwickelt, die domänenübergreifend gültige Gesamtstrukturen von Texten lernt, um die Robustheit bestimmter Analysen zu erhöhen. Die vier Teilansätze werden in Abschnitt 3 skizziert.

Die Korrektheit und Komplexität aller Teilansätze wurde, soweit möglich, formal analysiert. Mithilfe prototypischer Java-Implementierungen wurden darüber hinaus zahlreiche empirische Experimente auf neuen und auf anerkannten Benchmark-Datensätzen durchgeführt, um die erreichte Effektivität, Effizienz und Robustheit im Vergleich zu bestehenden Ansätzen zu evaluieren. Dabei wurden wissenschaftlich und industriell wichtige Textanalysen betrachtet, wie die Erkennung von Umsatzprognosen in News-Artikeln oder die Stimmungsanalyse von Reviews. Auch wurden Software-Frameworks und -Werkzeuge bereitgestellt, welche die praktische Anwendbarkeit der Ansätze demonstrieren.

Die in Abschnitt 4 besprochenen Ergebnisse der Dissertation belegen, dass anfragespezifische Pipelines in Echtzeit automatisch erstellt werden können. Stets wird eine optimale Ausführung erreicht, bei der Texte nur so weit analysiert werden, wie für die Ermittlung relevanter Informationen zwingend nötig. Durch die Reihenfolge-Optimierung lässt sich

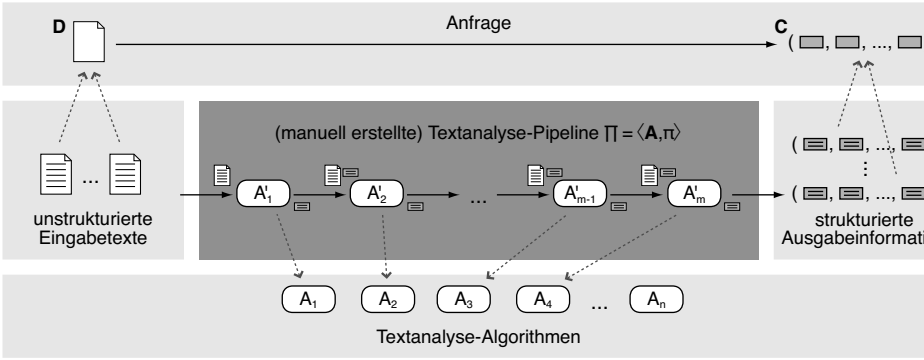

Abb. 1: Bisheriger Stand der Technik: In der manuell erstellten Textanalyse-Pipeline Π analysieren Algorithmen alle Eingabetexte vollständig, um die gesuchten Ausgabeinformationen zu produzieren.

die Effizienz von Pipelines um teils mehr als eine Größenordnung verbessern, ohne ihre Effektivität zu beeinflussen. Sogar auf heterogenen Texten wird die Effizienz erhalten, indem die Reihenfolge individuell angepasst wird. Auch gelingt durch den Fokus auf die Gesamtstruktur von Texten Robustheit in Analysen, die mit der Argumentation von Texten zu haben. Insgesamt verkörpert der entwickelte Ansatz damit einen wichtigen Baustein für den Einsatz beliebiger Pipelines in Suchmaschinen und Big Data Analytics.

2 Problemstellung

Ziel einer Textanalyse ist die Ermittlung strukturierter Ausgabeinformationen aus einer Menge unstrukturierter Eingabetexte $\mathbf{D}$. Dabei kann $\mathbf{D}$ eine Handvoll homogener Texte genauso wie ein potentiell nie endender Strom heterogener Texte sein. Die Ausgabeinformationen können zum Beispiel in Datenbanken gespeichert oder in Analytics-Anwendungen weiterverarbeitet verden. Enstprechend sind die Typen der Informationen vorgegeben, hier spezifiziert als geordnete Menge $\mathbf{C}$. Ein Typ kann etwa bestimmte Entitäten repräsentieren, wie Firmennamen oder Zeitausdrücke, oder auch Relationen, wie die Gründung einer Firma in einem bestimmten Jahr. Ein Paar $(\mathbf{D},\mathbf{C})$ lässt sich als Anfrage interpretieren, deren Ziel es ist, alle zu $\mathbf{C}$ passenden Ausgabeinformationen aus $\mathbf{D}$ zu ermitteln.

Im Fokus stehen im vorliegenden Beitrag Textanalyse-Pipelines; sie werden standardmäßig zur Bearbeitung einer gegebenen Anfrage eingesetzt und zuvor von Experten manuell erstellt. Konzeptuell lässt sich eine Pipeline als ein Tupel $\Pi = \langle \mathbf{A}, \pi \rangle$ modellieren. Dabei ist $\mathbf{A} = \{A'_1, \ldots, A'_m\}$ eine Menge von $m \geq 1$ Textanalyse-Algorithmen ist, die einer Grundmenge $\{A_1, \ldots, A_n\}$, $n \geq m$, entnommen wurde. π ist hingegen ein Schedule, der die Ausführungsreihenfolge der Algorithmen in $\mathbf{A}$ vorgibt. Jeder Algorithmus $A'_i \in \mathbf{A}$ analysiert die Texte aus $\mathbf{D}$, um Informationen bestimmter Typen aus $\mathbf{C}$ zu produzieren. Dafür benötigt er eine (potentiell leere) Menge an Typen als Eingabe, die von vorangehenden Algorithmen zu produzieren sind. Enstprechend muss π sicherstellen, dass die Abhängigkeiten zwischen Algorithmen erfüllt werden. Abb. 1 illustriert alle beschriebenen Konzepte; sie repräsentieren den Ausgangspunkt der Untersuchungen in diesem Beitrag.

Prinzipiell können mit Pipelines beliebige Textanalyse-Anfragen bearbeitet werden. In der Regel werden die von einer Pipeline produzierten Ausgabeinformationen nicht immer korrekt sein, da sie aus der Analyse natürlichsprachiger Texte resultieren, welche im Allgemeinen mehrdeutig sind. Vielmehr erzielt eine Pipeline eine bestimmte Effektivität, quantifiziert zum Beispiel als Anteil korrekter Ausgabeinformationen. Generell betrachtet ist die Erreichung hoher Effektivität das oberste Ziel einer Textanalyse. Insbesondere Suchmaschinen und Big-Data-Analytics-Anwendungen zielen darüber hinaus darauf ab, Informationen unmittelbar in Folge einer Anfrage zu ermitteln und zwar aus enorm großen Mengen heterogener Texte. Wie in Abschnitt 1 aufgezeigt, ist der Einsatz von Pipelines für entsprechende Anfragen bislang wegen fehlender Automatisierung, Effizienz und Robustheit noch weitgehend beschränkt. Um diese Probleme zu adressieren, verfolgt der im vorliegenden Beitrag präsentierte Ansatz drei Ziele:

1. *Die automatische Erstellung von Pipelines.* Dadurch lassen sich relevante Informationen ad-hoc für gegebene Anfragen ermitteln.
2. *Die Optimierung der Effizienz von Pipelines.* Dadurch lassen sich Analysen bei gleichem Zeitaufwand auf größeren Mengen von Texten durchführen.
3. *Die Optimierung der Robustheit von Pipelines.* Dadurch lässt sich die Effektivät von Analysen auf heterogenen Texten verbessern.

Stand der Technik In einigen Fällen liefern führende Suchmaschinen bereits heute direkt relevante Informationen, etwa bei der Suche nach bekannten Personen oder anderen Entitäten. Zum Teil werden solche Informationen mittels Textanalysen gefunden [Pa11], zum Teil entstammen sie aber auch frei verfügbaren Wissensbasen wie www.freebase.com. So oder so ist die Ermittlung der Informationen bislang auf antizipierte Anfragen beschränkt, da Suchmaschinen die Informationen in einem vorberechneten Index verwalten.

Für die genannten Ziele existiert ansonsten noch keine Gesamtlösung. Einige verwandte Arbeiten adressieren zumindest einzelne Ziele in abgewandelter Form. So gibt es Ansätze, die anstelle einer anfragespezifischen Pipeline eine lediglich durch die Anfrage konkretisierte Analyse verwenden [Ba07], was sich allerdings nur für grundlegende Informationtypen machen lässt. In der Analytics-Anwendung SystemT beschreibt ein Nutzer alternativ dazu die durchzuführenden Analysen, während die Umsetzung automatisch geregelt wird [Kr09]. Ähnlich wie der hier vorgestellte Ansatz beschränkt SystemT Analysen auf potentiell relevante Textteile und optimiert ihre Reihenfolge hinsichtlich Effizienz, ist aber auf den Einsatz regelbasierter Algorithmen beschränkt. Das Effizienzproblem lässt sich umgehen, indem potentiell relevante Informationstypen bereits in den Suchindex aufgenommen werden [Ca05]. Das funktioniert natürlicherweise aber ebenfalls nur für antizipierte Typen. Auch durch eine Beschränkung auf einfache, skalierbare Analysen wird Effizienz erreicht, was jedoch in der Regel mit Effektivitätsverlust einhergeht [PRH04]. Für Robustheit, schließlich, ist der meist verbreitete Ansatz Domänenanpassung [BDP07], wobei diese mithilfe von Beispieltexten aus den Zieldomänen geschieht. Nicht zuletzt bei Suchmaschinen sind die Zieldomänen jedoch nicht vorhersagbar. Daher zielt der vorgestellte Ansatz darauf ab, direkt Domänenunabhängigkeit zu erreichen. Für andere als die hier untersuchten Textanalysen gibt es vergleichbare Ansätze, etwa [MC11].

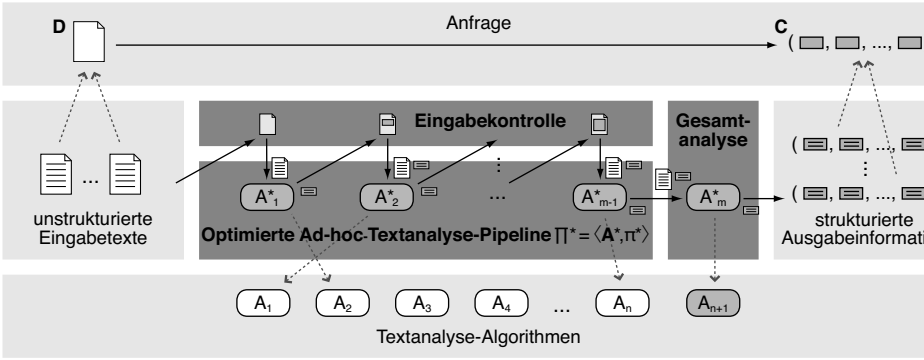

Abb. 2: Überblick über den vorgestellten Ansatz: Jeder Textanalyse-Algorithmus A_i^* der ad-hoc erstellten Pipeline $\Pi^* = \langle \mathbf{A}^*, \pi^* \rangle$ erhält nur Teile eines Eingabetextes von der Eingabekontrolle, für die seine Ausgabe relevant ist. Der Schedule π^* der Algorithmen ist effizienzoptimiert. Die Effektivität der Ausgabeinformationen wird mittels einer Gesamtanalyse domänenübergreifend erhalten.

3 Ansatz

Um es Suchmaschinen und Big-Data-Analytics-Anwendungen zu ermöglichen, Texte ad-hoc zu analysieren, wird ein Ansatz vorgeschlagen, der im Kern auf drei Ideen basiert:

1. *Optimierte Ad-hoc-Textanalyse-Pipelines.* Die in einer Pipeline zu verwendenden Algorithmen lassen sich aus einer Anfrage automatisch ableiten. Ihr Schedule lässt sich anhand ihrer Laufzeiten und produzierten Ausgabeinformationen optimieren.

2. *Eingabekontrolle.* Die Teile eine Textes, die ein Algorithmus analysieren muss, lassen sich von der Anfrage, der Typen seiner Ausgabeinformationen und der bislang produzierten Ausgabeinformationen ableiten.

3. *Gesamtanalyse.* Die Robustheit bestimmter Analysen lässt sich erhöhen, indem ihr Fokus auf die Gesamtstruktur eines Textes anstelle seines Inhalts gelegt wird.

Abb. 2 zeigt, wie sich der bisherige (in Abb. 1 dargestellte) Stand der Technik auf Grundlage dieser Ideen weiterentwickeln lässt. Zur Umsetzung der Ideen wurden in [Wa15] vier Teilansätze erarbeitet, die im Folgenden skizziert werden.

Ad-hoc-Textanalyse-Pipelines Damit sich auch unvorhergesehene Anfragen ad-hoc beantworten lassen, muss eine Textanalyse-Pipeline unmittelbar erstellt werden können. Zu diesem Zweck wurde eine Formalisierung der prozessorientierten Sicht auf Textanalyse erarbeitet. Konkret wird jeder Algorithmus als Aktion aufgefasst, die – falls anwendbar – den Zustand eines Eingabetextes (im Sinne der bislang im Text gefundenen Informationen) in einen anderen Zustand (erweitert um die Ausgabeinformationen des Algorithmus) überführt. Diese Sicht ist konsistent zu Standard-Frameworks für Textanalyse, wie Apache UIMA. Sie erlaubt, die Ad-hoc-Erstellung als Suche nach einer zulässigen Folge von Aktionen und damit als Planungsproblem aufzufassen. In [Wa15] wird die Erstellung mittels der Künstliche-Intelligenz-Technik *Partial Order Planning* adressiert [RN09], wobei verschiedene Effizienz- und Effektivitätskriterien priorisiert werden können.

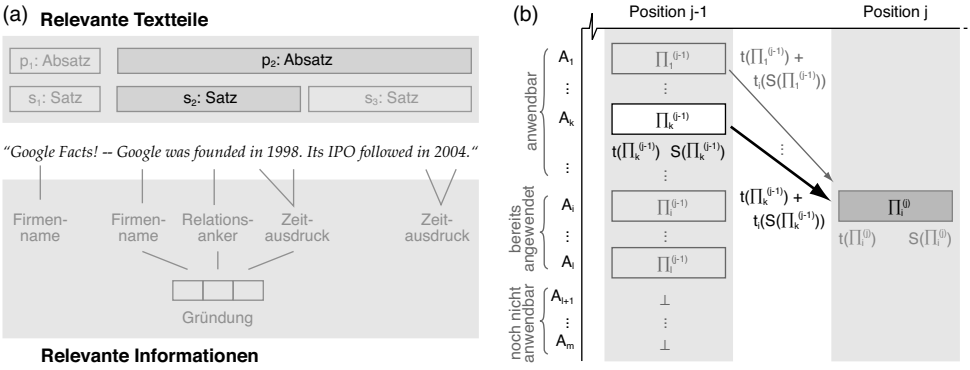

Abb. 3: (a) Gegenüberstellung der relevanten Teile eines Textes und der darin befindlichen relevanten Informationen bezüglich einer Relation *Gründung*. (b) Bestimmung der laufzeitoptimalen Pipeline $\Pi_i^{(j)}$ mit Algorithmus A_i in Position j auf Basis aller Pipelines der Länge $j-1$.

Eingabekontrolle Anstatt jeden Eingabetext direkt von einem zum nächsten Algorithmus in einer Pipeline weiterzugeben, wird eine Eingabekontrolle eingeführt, die für jeden Algorithmus entscheidet, welche Teile des Textes er zu analysieren hat. Dafür wird nach jeder Algorithmusausführung ermittelt, welche Teile noch alle zur Beantwortung einer Anfrage relevanten Informationen enthalten können. Abb. 3(a) illustriert die relevanten Teile eines Beispieltextes auf Satz- und Absatzebene. Wie in [Wa15] gezeigt, eignet sich dafür *Assumption-based Truth Maintenance* [RN09]. Dazu wird die Relevanz jedes Textteils als aussagenlogische Formel modelliert. Schlussfolgerungsverfahren ermöglichen dann, jeden Algorithmus nur auf für ihn relevanten Teilen auszuführen. Dadurch lässt sich jegliche unnötige Analyse vermeiden. Zusätzlich lässt sich der Tradeoff zwischen Effizienz und Effektivität beeinflussen, indem kleinere Textteile betrachtet werden, was Zeit spart, aber unter Umständen manche Informationen nicht mehr auffindbar macht.

Optimierte Textanalyse-Pipelines Unter Verwendung einer Eingabekontrolle entscheidet der Schedule der verwendeten Algorithmen über die Laufzeiteffizienz einer Pipeline. Ein optimales Scheduling erweist sich als dynamisches Programmierproblem, wie in Abb. 3(b) illustriert. Dort resultiert die Laufzeit $t(\Pi_i^{(j)})$ einer Pipeline $\Pi_i^{(j)}$ der Länge j aus der minimalen Summe der Laufzeit $t(\Pi_k^{(j-1)})$ einer Pipeline der Länge $j-1$ und der Laufzeit $t_i(S(\Pi_k^{(j-1)}))$ von Algorithmus A_i auf dem noch relevanten Textteil $S(\Pi_k^{(j-1)})$. In der Praxis sind die Laufzeiten und relevanten Textteile im Vorhinein nicht bekannt. Stattdessen kann die optimale Pipeline aber auf einer Trainingsmenge an Texten mit *informierter A*-Suche* [RN09] approximiert werden. Probleme treten nur auf, wenn sich Eingabetexte als sehr heterogen bezüglich der Verteilung relevanter Informationen herausstellen, da es dann keinen stets optimalen Schedule gibt. Um damit umzugehen, wird in [Wa15] ein *selbstüberwachter maschineller Lernansatz* [Ba07] entwickelt, der die Laufzeiten verschiedener Schedules für jeden gegebenen Text vorhersagt und die Pipeline passend adaptiert.

Gesamtanalyse Die vorgeschlagene Gesamtanalyse dient schließlich dazu, die Domänenrobustheit einer Pipeline durch einen Fokus auf die Gesamtstruktur von Texten zu erhöhen,

während vom Inhalt der Texte abstrahiert wird. Entsprechend Abb. 2 lässt sich die Gesamtanalyse als alternativer letzter Algorithmus einer Pipeline verstehen. Er zielt speziell auf Analysen ab, die mit der Klassifikation argumentativer Texte (wie Reviews oder Essays) zu tun haben. Mittels einer *überwachten Clustering-Variante* können in Trainingstexten domänenübergreifend gültige Muster in Argumentationsverläufen bestimmt werden [Wa15]. Durch den Vergleich des Verlaufs eines Textes mit allen Mustern lässt sich dann die Gesamtstruktur des Textes in bisher nicht dagewesener Weise messbar machen.

4 Ergebnisse

Der beschriebene Ansatz zielt darauf ab, das Design von Textanalyse-Pipelines zu automatisieren, die Effizienz ihrer Ausführung zu optimieren und die Robustheit ihrer Ausgabe zu verbessern. Um den Erfolg des Ansatzes zu überprüfen, wurden im Rahmen der Dissertation [Wa15] einzelne Aspekte formal analysiert, vor allem aber alle Teilansätze in zahlreichen Experimenten auf neu entwickelten und auf anerkannten Benchmark-Datensätzen empirisch evaluiert. Im Folgenden werden wesentliche Ergebnisse und daraus gewonnene Erkenntnisse anhand der drei adressierten Probleme zusammengefasst:

Automatische Erstellung von Pipelines Die Ad-hoc-Erstellung von Textanalyse-Pipelines wurde für insgesamt 14 Anfragen verschiedener Komplexität evaluiert, welche die Erkennung von Marktprognosen sowie die Erkennung biomedizinischer Relationen betreffen. Die Ergebnisse legen nahe, dass sich effektive und effiziente Pipelines in realistischen Situationen nahezu in Nullzeit automatisch erstellen lassen. So dauerte die Erstellung von Pipelines mit bis zu 21 Algorithmen aus einer Grundmenge von 76 Algorithmen auf einem Standardrechner maximal 26 Millisekunden (gemittelt über 25 Durchläufe). Nach bestem Wissen wird dadurch zum ersten Mal ermöglicht, Textanalysen für unvorhergesene Anfragen ad-hoc durchzuführen. Offene Probleme resultieren im Wesentlichen aus der Automatisierung; so lässt sich die Erstellung zwar auf verschiedene Priorisierungen hin ausrichten (etwa “Effizienz über Effektivität”), Gewichtungen erscheinen hingegen schwierig. Auch kann das Zusammenspiel verschiedener Algorithmen in Ad-hoc-Szenarien natürlich im Vorhinein nicht überprüft werden.

Optimierung der Effizienz von Pipelines Die Eingabekontrolle adressiert eine Schwachstelle nahezu aller existierenden Textanalysen; sie erkennt auf formale Weise im Vorheinen, auf welchen Teilen eines Textes eine Analyse irrelevant ist, und erlaubt so die Maximierung der Effizienz einer gegebenen Pipeline. Das entstehende Optimierungspotential lässt sich nicht generell quantifizieren, da es von den verwendeten Algorithmen und der Menge relevanter Informationen in Texten abhängt. In mehreren Fallstudien lag der Anteil des von einem Algorithmus im Durchschnitt verarbeiteten Textes aber stets zwischen 40% und 80%. Exemplarisch zeigt Abb. 4 die entsprechend analysierten Anteile für eine Pipeline zur Erkennung der bereits erwähnten Gründungsrelation auf dem englischsprachigen CoNLL-2003-News-Datensatz [TM03]. In Fällen, wo die meisten Analysen alle Textteile betreffen (wie bei vielen Textklassifikationsproblemen), wird der Effekt der Eingabekontrolle hingegen gering sein. Der zusätzliche zeitliche Aufwand lag in allen Experimenten aber nur bei 1% der Gesamtlaufzeit und ist daher im Grunde vernachlässigbar. Auch sonst

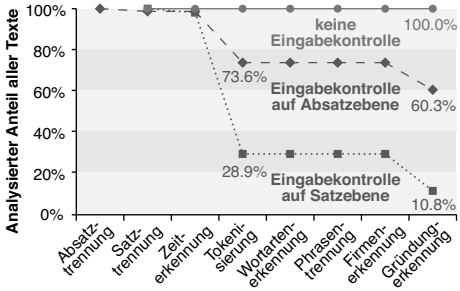

Abb. 4: Analysierter Anteil des CoNLL-2003-Datensatzes für jeden Algorithmus einer Pipeline zur Erkennung von Gründungsrelationen.

Ansatz	Scheduling (20 Texte)	Analyse (366 Texte)
Standard-Pipeline	–	61.8 s ±1.4 s
Greedy Scheduling	–	30.4 s ±1.4 s
1-best A*-Suche	11.4 s ±0.7 s	19.6 s ±0.9 s
10-best A*-Suche	43.1 s ±0.6 s	18.4 s ±0.3 s
20-best A*-Suche	109.9 s ±3.2 s	15.8 s ±0.4 s
100-best A*-Suche	144.9 s ±4.3 s	15.6 s ±0.5 s
Optimaler Schedule	–	15.5 s ±0.3 s

Tab. 1: Laufzeit von sechs Ansätzen zum Scheduling einer Pipeline für Umsatzprognosen auf 20 Trainings- und ihrer Analyse von 366 Testtexten.

birgt die Eingabekontrolle keine nennenswerten Nachteile, weswegen sie für die Zukunft eine logische Erweiterung von Standard-Frameworks wie Apache UIMA darstellt.

Bei gegebener Eingabekontrolle entscheiden die irrelevanten Teile von Texten über den optimalen Schedule einer Pipeline, wie formal bewiesen wurde; relevante Teile müssen ohnehin von allen Algorithmen analysiert werden. Auf homogenen Texten wurde der Schedule zuverlässig mittels des entwickelten A*-Suchansatzes optimiert. Tab. 1 zeigt Ergebnisse eines Beispielexperiments mit zehn Algorithmen zur Erkennung von Umsatzprognosen für vier A*-Konfigurationen auf 20 Trainingstexten. Die Ergebnisse verdeutlichen den Nutzen des Schedulings: Gegenüber einer Standard-Pipeline mit Eingabekontrolle wurde die Laufzeit etwa um Faktor 4 verbessert. Dabei wurde nahezu die optimale Laufzeit erreicht. Der zusätzliche Aufwand lohnt sich jedoch nur auf großen Datenmengen: Auf den 366 Testtexten ist ein Greedy-Scheduling-Ansatz, der einzig auf Laufzeitschätzungen der Algorithmen basiert, noch ebenbürtig. Gegenüber Pipelines ohne Eingabekontrolle konnte die Laufzeit sogar um bis zu Faktor 16,5 verbessert werden. Auf heterogenen Texten reicht ein fester Schedule mitunter hingegen nicht aus; dort wurde die Laufzeit des optimierten Schedules vom entwickelten Lernansatz für adaptives Scheduling noch um bis zu 30% verringert. Eine Abschätzung der Heterogenität erwies sich jedoch als notwendig, da Vorhersagefehler des Lernansatzes auf homogenen Texten zu Nachteilen führen können.

Optimierung der Robustheit von Pipelines Die Gesamtanalyse wurde auf Reviews unterschiedlicher Domänen evaluiert. Es wurden Muster gefunden, die typische Verläufe von Stimmungen und den in Reviews auftretenden Diskursrelationen repräsentieren. Sie geben neue Einblicke, wie Menschen heutzutage argumentieren. Maschinelle Lernexperimente zur Stimmungsanalyse auf einem neuen Hotelreview-Datensatz und einem Standard-Filmreview-Datensatz [PL05] untermauern den Nutzen der Muster zur Erreichung von Robustheit: Ihre Effektivität blieb in acht domänenübergreifenden Experimenten fast immer stabil; im Durchschnitt fiel der Anteil korrekt klassifizierter Stimmungen um lediglich sieben Prozentpunkte. Im Vergleich dazu betrug der Verlust auf Basis der Verteilung der Stimmung in Reviews 12 Prozentpunkte und auf Basis des Inhalts gar 23 Prozentpunkte. Dennoch bleibt die insgesamt erreichte Effektivität weiter verbesserbar. Eine Kombination

mit Verfahren zur Domänenanpassung erscheint in dieser Hinsicht vielversprechend, wird aber als Arbeit für zukünftige Forschung offen gelassen.

Insgesamt wurde mit den gewonnenen Erkenntnissen ein zentraler Grundstein für den Ad-hoc-Einsatz von Textanalyse-Pipelines auf großen Datenmengen gelegt. Um diese Forschung fortführen und die durchgeführten Experimente reproduzieren zu können, wurden alle wesentlichen Teilansätze als Open-Source-Software umgesetzt. Auch wurden drei Benchmark-Datensätze für wissenschaftlich und industrielle relevante Textanalysen veröffentlicht. Software und Daten sind auf http://is.upb.de/?id=wachsmuth frei zugänglich.

5 Fazit

Textanalyse-Pipelines ermöglichen, relevante Informationen für komplexe Anfragen in natürlichsprachigen Texten zu finden. Bislang ist ihr Einsatz in Suchmaschinen und Big-Data-Analytics-Anwendungen jedoch begrenzt, da es an Automatisierung, Effizienz und Robustheit mangelt. Dieser Beitrag hat einen Überblick über den in [Wa15] entwickelten Ansatz zur Adressierung dieser Probleme und die dort erreichten Ergebnisse gegeben. Mittels klassischer und moderner Techniken der künstlichen Intelligenz werden Pipelines automatisch erstellt, bezüglich ihrer Laufzeiteffizienz optimiert und hinsichtlich ihrer Effektivität auf vielfältigen Texten verbessert. Empirische Evaluierungen belegen, dass der Ansatz beliebige Textanalysen ad-hoc ermöglicht, ihren Einsatz auf deutlich größeren Datenmengen zulässt und für viele Anfragen robustere Analysen gewährleistet. Zwar ist der Ertrag einiger Teilansätze in manchen Fällen nur gering. Auch bleibt das generelle Effektivitätsproblem der Verarbeitung natürlicher Sprache bestehen. Dennoch stellen die erzielten Ergebnisse einen wichtigen Baustein hin zum Einsatz von Pipelines in Suchmaschinen und Big Data Analytics dar. Ob sich die zukünftige Informationssuche tatsächlich in diese Richtung bewegt, wird sich herausstellen. Die unmittelbare Suche nach relevanten Informationen hat jedenfalls bereits begonnen [Et11, Pa11, KH13].

Literaturverzeichnis

[Ba07] Banko, Michele; Cafarella, Michael J.; Soderland, Stephen; Broadhead, Matt; Etzioni, Oren: Open Information Extraction from the Web. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence. S. 2670–2676, 2007.

[BDP07] Blitzer, John; Dredze, Mark; Pereira, Fernando: Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics. S. 440–447, 2007.

[Ca05] Cafarella, Michael J.; Downey, Doug; Soderland, Stephen; Etzioni, Oren: KnowItNow: Fast, Scalable Information Extraction from the Web. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing. S. 563–570, 2005.

[Et11] Etzioni, Oren: Search Needs a Shake-up. Nature, 476:25–26, 2011.

[HP10] HP Labs: Annual Report. Bericht, 2010. Verfügbar auf http://www.hpl.hp.com/news/2011/jan-mar/pdf/HPL_AR_2010_web.pdf (Zugriff am 2. Februar 2016).

[Ka10] Kano, Yoshinobu; Dorado, Ruben; McCrohon, Luke; Ananiadou, Sophia; Tsujii, Jun'ichi: U-Compare: An Integrated Language Resource Evaluation Platform Including a Comprehensive UIMA Resource Library. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation. S. 428–434, 2010.

[KH13] Kelly, John E.; Hamm, Steve: Smart Machines: IBM's Watson and the Era of Cognitive Computing. Columbia University Press, New York, NY, USA, 2013.

[Kr09] Krishnamurthy, Rajasekar; Li, Yunyao; Raghavan, Sriram; Reiss, Frederick; Vaithyanathan, Shivakumar; Zhu, Huaiyu: SystemT: A System for Declarative Information Extraction. SIGMOD Records, 37(4):7–13, 2009.

[MC11] Menon, Rohith; Choi, Yejin: Domain Independent Authorship Attribution without Domain Adaptation. In: Proceedings of the International Conference Recent Advances in Natural Language Processing 2011. S. 309–315, 2011.

[Pa11] Pasca, Marius: Web-based Open-Domain Information Extraction. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management. S. 2605–2606, 2011.

[PL05] Pang, Bo; Lee, Lillian: Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. S. 115–124, 2005.

[PRH04] Pantel, Patrick; Ravichandran, Deepak; Hovy, Eduard: Towards Terascale Knowledge Acquisition. In: Proceedings of the 20th International Conference on Computational Linguistics. S. 771–777, 2004.

[RN09] Russell, Stuart J.; Norvig, Peter: Artificial Intelligence: A Modern Approach. Prentice-Hall, Upper Saddle River, NJ, USA, 3rd. Auflage, 2009.

[Sa08] Sarawagi, Sunita: Information Extraction. Foundations and Trends in Databases, 1(3):261–377, 2008.

[TM03] Tjong Kim Sang, Erik F.; Meulder, Fien De: Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003. S. 142–147, 2003.

[Wa15] Wachsmuth, Henning: Text Analysis Pipelines—Towards Ad-hoc Large-scale Text Mining. Lecture Notes in Computer Science 9383. Springer, 2015.

Henning Wachsmuth wurde 1983 in DeKalb, Illinois geboren und wuchs danach in Osnabrück und Bielefeld auf. Nach dem Abitur am Bielefelder Helmholtz-Gymnasium im Jahr 2002 und dem darauf folgenden Zivildienst studierte er ab 2003 an der Universität Paderborn Informatik mit Nebenfach Medienwissenschaft. Dort erhielt er 2006 zunächst den Bachelor, 2009 schloss er sein Studium dann als Diplom-Informatiker ab. Anschließend arbeitete er am s-lab – Software Quality Lab der Universität Paderborn mit Schwerpunkt in der algorithmischen Verarbeitung natürlichsprachiger Texte. Zeitgleich promovierte er am Lehrstuhl Datenbank- und Informationssysteme und verteidigte Anfang 2015 erfolgreich seine Dissertation. Seit April 2015 forscht er als Postdoc am Lehrstuhl Content Management und Web-Technologien der Bauhaus-Universität Weimar an der computerlinguistischen Analyse von Argumentation.