

Protokolle für die Sitzung vom 11.05.2015

13. Mai 2015

Präsentationen

Andreas, dessen Arbeit sich mit den Auswirkungen von Negation in *sentiment analysis* befasst, gab zu Beginn seines Vortrags einige grundlegende und allgemeine Erläuterungen zu den zentralen Begriffen *sentiment* und *Negation* und zur Methode der *Support Vector Machines* (SVM). Das schwer übersetzbare Wort 'sentiment' bezeichnet die wertende Polarität eines Ausdrucks, seine Stimmung, seine emotionalen Implikationen; in Andreas' Arbeit kann das sentiment positiv, negativ oder neutral sein. Ein problematisches Beispiel wurde gleich zu Beginn von Prof. Schulz kritisiert: der Satz 'I've never seen something like this' kann in verschiedenen Kontexten verschiedene sentiments ausdrücken. Andreas arbeitet jedoch allein auf Satzbasis, sodass das Beispiel als neutral eingestuft wird.

Zur Negation wurde erwähnt, dass sie in der Regel durch einen Negator eingeleitet wird (z.B. 'nicht'), in dessen Skopus das Argument liegt, welches von ihm verneint wird. Die Größe des Skopus zu bestimmen wird eine der Hauptaufgaben der Arbeit sein.

Was sind nun SVM? Anhand eines Beispiels mit zwei Features (Wortlänge, Doppelbuchstaben) erklärte Andreas, wie aus den Features ein Vektor erstellt wird: 'good', 'happy' und 'bad' hätten die entsprechenden Vektoren (4, True), (5, True) und (3, False). Die Features werden mit den gelabelten Trainingsdaten (Goldstandard) an die SVM übergeben, welche daraus 'lernt', welche Merkmale aussagekräftig sind. Da 'good' und 'happy' beide True als zweiten Wert haben und beide im Goldstandard als positiv gelabelt sind, 'bad' dagegen negativ und an dieser Stelle ein False hat, würde die SVM fälschlich Doppelbuchstaben als ein relevantes Merkmal positiver Wörter einstufen. Indem man möglichst viele Daten und sinnvollere Features verwendet, versucht man derlei Zufälligkeiten auszuschalten und eine aussagekräftige SVM zu trainieren.

Womit wir bei der eigentlichen Arbeit wären. Andreas verwendet auf sentiment gelabelte Twitterdaten und zwar etwa 10000 Tweets (8000 Training-, 2000 Test-Set). Davon sind etwa 60% neutral, 20% positiv und 20% negativ, sodass das Programm nicht durch unausgewogene Trainingsdaten ungenaue Ergebnisse liefert. Das Baseline System liefert einen F-Score von ca. 57, ohne dass spezielle Negationsfeatures implementiert wurden, etwa 58,5 mit diesen Features. Vor allem bei den Tweets mit Negation kann man mit den Features eine deutliche Verbesserung erreicht werden, bei den anderen zumindest keine Verschlechterung. Prof. Schulz wies darauf hin, dass Precision und Recall nicht unbedingt die unbestritten besten Metriken seien, da z.B. bei einem positiven Satz, negativ und neutral undifferenziert als falsch gewertet würden, während man ja neutral durchaus auch als 'weniger falsch' werten könnte. Mit weiteren Features versucht Andreas dieses Baseline System weiterhin zu verbessern. Wie schon angedeutet, könnte die Un-

tersuchung der Skopuseigenschaften von Negatoren entscheidende Hinweise geben, z.B. dass der Skopus häufig bei einem Wort wie 'but' oder 'however' endet, da hier ein neuer semantischer Abschnitt beginnt, der dem ersten (negierten) Abschnitt entgegengesetzt ist. Zudem stellte Andreas zwei Hypothesen vor, welche die Auswirkungen des Negators betreffen: die erste besagt, dass der Negator eine reine Invertierung des Arguments bewirkt, während die sogenannte Shift-Hypothese unterschiedlich starke Auswirkungen auf den Polaritätswechsel und auch auf den Negator selbst annimmt.

Die Bachelorarbeit wird von Dr. Thang Vu betreut.



Die Arbeit wird von Dr. Leiss betreut und befasst sich mit der Grammatik und Auswertung von Zeit- und Ortsadverbialien. Sie ist im Grunde eine Erweiterung des allseits bekannten, in Prolog geschriebenen Datenbankabfragesystems, welches Fragen aus dem Bereich der Astronomie in natürlicher Sprache beantworten kann. Somit kann das mühsame Erlernen von query-languages für die Benutzer vermieden werden. Zu diesem Zweck wird die Frage ge-parsed und ein Syntaxbaum erzeugt; aus den Semantiken der einzelnen Konstituenten wird dann die Semantik der ganzen Frage in Form einer prädikatenlogischen Formel errechnet, mit der die Datenbank abgefragt wird. Schließlich wird eine natürlichsprachliche Antwort mit den gewünschten Daten erzeugt. Besonders Fragen, die komplexe Zusammenhänge und unspezifische Teilfragen betreffen, lassen sich mit der logischen Notation effizient darstellen: 'Welcher Astronom entdeckte einen Mond, der einen Planeten umkreist?' würde übersetzt in $qu(x, astronom(x) \wedge mond(y) \wedge entdecken(x, y) \wedge planet(z) \wedge umkreisen(y, z))$, was eindeutige Antworten in relativ kurzer Zeit erlaubt.

Zur Implementierung von Zeit- und Ortsadverbialien gibt es verschiedene Ansätze, welche Martin anhand vom Beispiel 'Piazzi entdeckte Ceres im Jahr 1801' vorstellte:

- als zusätzliches Argument: $entdecken(Piazzi, Ceres, 1801)$
- als zusätzliches Prädikat: $imJahr1801(entdecken(Piazzi, Ceres))$
- erst in der Auswertung: $[entdecken(Piazzi, Ceres)]_{1801}$

Die erste Lösung ist offenkundig am einfachsten, da sich der Mehraufwand auf die Einführung neuer Argumente beschränkt; Verschachtelungen und komplexe Auswertungsverfahren sind hier nicht nötig. Da das Ziel der Arbeit darin besteht, eine möglichst flexible Grammatik für unterschiedliche Anwendungen zu erstellen, wird dieser Ansatz ausgewählt. Um die Grammatik möglichst einfach und adaptierbar zu halten wird die Genauigkeit und Art von Zeit- und Ortsangaben übrigens nicht in der Grammatik festgehalten, sondern in der Datenbank selbst. Im Falle der Astronomie-Datenbank werden auf Zeitanfragen nur Jahreszahlen angegeben, als Ortsangaben dienen Koordinaten auf Sternenkarten. Martin zeigte ein Beispiel, in dem er seine Grammatik auf eine Bibliotheksdatenbank übertrug und problemlos zeitliche Daten auf Tage und Stunden umstellen konnte und räumliche Informationen z.B. auf Bücherregale; dabei ändern sich nur die Definitionen in der Datenbank, nicht die Grammatik.

Außerdem erklärte Martin die Bedeutung von Anzahlquantoren im Zusammenhang von Zeitangaben, z.B. in der Phrase 'Zweimal im Jahr 1801', wo das Adverb 'Zweimal' als logischer Quantor aufgefasst wird.

Der Vortrag von Fabian fand in den letzten 10 Minuten der Sitzung statt und wurde daher nicht so ausführlich diskutiert wie die beiden vorherigen. Sein Thema ist die Entwicklung von lokalen Grammatiken aus Medline Phrasen, wozu er den graphischen Editor von Unitex verwendet. Sein Fokus liegt auf Phrasen, die auf irgendeine Art Verursachung ausdrücken; Standardbeispiel ist *x causes y*. Auch dieser Vortrag wurde mit der Beantwortung einiger allgemeiner Fragen eingeleitet, hier mit der Erklärung lokaler Grammatiken: es sind Schemata, mit denen sich Sequenzen von Wörtern generieren lassen, die eine semantische Einheit bilden. Medline ist ein Korpus, das aus medizinischen Artikeln besteht; von dem sehr großen Korpus verwendet Fabian eine immer noch ziemlich große Teilmenge, deren Größe er mit ungefähr 120 MB angab. Tools von Unitex, die in der Arbeit genutzt werden, sind z.B. der Tokenisierer, Wortlisten zu bestimmten Themenbereichen und eben der Editor zur Erstellung von lok. Grammatiken. Der große Vorteil ist, dass Wortlisten eingebunden werden können, sodass eine mögliche Grammatik z.B. so aussehen würde

< Krankheit > → causes → < Symptom >

Die Begriffe in spitzen Klammern stehen hier für Listen von Begriffen, aus denen jedes Element ge-matched werden kann. Das Ziel der Arbeit ist es sprachliche Varianten der Verursachungs-Beziehung zu finden und lokale Grammatiken dafür zu entwickeln

Formal:

- * **übersichtliche Aufteilung**
- * **Name d. Referenten, BA- Betreuer und Thema**
- * **Rechtschreibung, Kommasetzung, Darstellung von Implementierung,...**

Inhaltlich:

- * **ergebnisorientiert und nachvollziehbar**
- * **ausreichende Erläuterungen**
- * **anschauliche Beispiele**
- * **Diskussion auf wesentliche Informationen beschränkt**