Digital Humanities: Übung 1

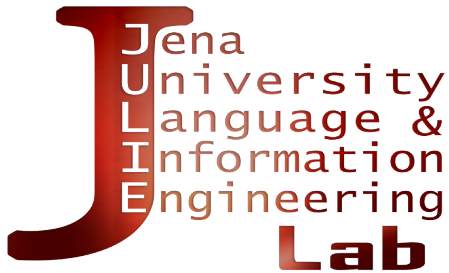
Suche mit Boolschen Operatoren und Regulären Ausdrücken

Sven Büchel

Jena Language & Information Engineering (JULIE) Lab
Friedrich-Schiller-Universität Jena, Germany

http://www.julielab.de

23. Mai 2017



Programm für heute

- Vorstellungsrunde
- Aufbau und Ziele der Übung
- Klärung des Vorlesungsinhalts (bei Bedarf)
- Erarbeitung formaler Konzepte
  - Aussagenlogik
  - Reguläre Ausdrücke
- Aufgaben zu nächster Woche

Sven Buechel

Sven Buechel

julielab.de/Staff/Sven+Buechel.html

67%

Home Students Contact Sitemap

» JULIE Lab » Staff » Sven Buechel

- Prof. Dr. Udo Hahn
- Dr. Sascha Schäuble
- Erik Fäßler
- Johannes Hellrich
- Franz Matthies
- Sven Buechel
- Christina Lohr
- Benjamin Reding
- Alumni

Sven Buechel

I joined Prof. Udo Hahn's Group in summer 2016 to follow up on the research associated with my Bachelor thesis on emotional language use by organizations. The focus of my work is both on the development of novel methodologies for measuring emotion in text as well as applying those methods in Computational Social Science and the Digital Humanities.

Selected Publications

Sven Buechel and Udo Hahn. 2017. EmoBank: Studying the Impact of Annotation Perspective and Representation Format on Dimensional Emotion Analysis. In ***EACL 2017*** - *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics. Valencia, Spain, April 3-7, 2017.* Volume 2, Short Papers, pages 578-585. Available: http://aclweb.org/anthology/E17-2092

Sven Buechel and Udo Hahn. 2017. A Flexible Mapping Scheme for Discrete and Dimensional Emotion Representations: Evidence from Textual Stimuli. In ***CogSci 2017*** - *Proceedings of the 39th Annual Meeting of the Cognitive Science Society. London, UK, July 26-29, 2017* [paper accepted].

Sven Buechel and Udo Hahn. 2016. Emotion analysis as a regression problem - Dimensional models and their implications on emotion representation and metrical evaluation. In ***ECAI 2016*** - *Proceedings of the 22nd European Conference on Artificial Intelligence. The Hague, The Netherlands, August 29 - September 2, 2016*, pages 1114-1122. Available: http://ebooks.iospress.nl/volumearticle/44864

Full Publication List

Teaching Experience

- Introduction to Computational Linguistics and Language Technology, Fall 2016.
- Computational Linguistics I, Fall 2016.
- Introduction to the Digital Humanties, Fall 2016, Spring 2017.

Contact

Sven Buechel
Research Assistant

Email	sven.buechel@uni-jena.de
Phone	+49 3641 9-44305
Postal Address	Fürstengraben 27 07743 Jena Germany
Room	E 008

letzte Änderung: am 13.05.201

Aufbau und Ziele der Übung

- Übung im unregelmäßigen Wechsel mit Vorlesung
- Klärung offener Fragen
- Vertiefte Auseindersetzung mit formalen Konzepten
- Praxisnahe Demonstration/Anwendung relevanter Technologien
- Übungsblätter im Anschluss an jede Sitzung
  - Mindestens 50% der Punkte für Prüfungszulassung!

Vorlesungsinhalt

Gibt es Fragen?

Grundlagen der Aussagenlogik

Überblick

- Aussagenlogik beschäftigt sich mit dem Wahrheitswert von Aussagen
- Komplexe Aussagen entstehen durch Verknüpfung von strukturlosen **Elementaraussagen** (**Atomen**) durch **Junktoren** (**Boolesche Operatoren**)
- Natürlichsprachliches Beispiel:
  Aussage A ist wahr, Aussage B ist falsch, damit ist die komplexe Aussage "A und B" falsch

Boolesche Operatoren

- Es gibt genau zwei zulässige **Wahrheitswerte** für Aussagen:
  - wahr: 1
  - falsch: 0
- Elementaraussagen werden per Konvention mit Kleinbuchstaben (a, b, c,...) dargestellt. Diese werden als Variablen behandelt.
- Verknüpfungen / Operatoren zwischen Variablen
  - `NICHT` `!` / `NOT` $\neg a$ dreht Wahrheitswert um
  - `UND` `&&` $a \wedge b$ vgl. mit Multiplikation $*$
  - `ODER` `||` $a \vee b$ vgl. mit Addition $+$

Wahrheitswert-Tabellen

- Zur Bestimmung des Wahrheitswertverlaufs logischer Aussagen
- Zeigt den Wahrheitswert einer komplexen Aussage in Abhängigkeit des Wahrheitswerts der logischen Atome)

		NICHT *a*	NICHT *b*	*a* ODER *b*	*a* UND *b*
a	b	$\neg a$	$\neg b$	$a \vee b$	$a \wedge b$
0	0	1	1	0	0
1	0	0	1	1	0
0	1	1	0	1	0
1	1	0	0	1	1

Rechenregeln für Boolesche Operatoren

1.	$a \wedge b = b \wedge a$ $a \vee b = b \vee a$	$a * b = b * a$ $a + b = b + a$
2.	$(a \wedge b) \wedge c = a \wedge (b \wedge c)$ $(a \vee b) \vee c = a \vee (b \vee c)$	$(a * b) * c = a * (b * c)$ $(a + b) + c = a + (b + c)$
3.	$a \wedge a = a$ $a \vee a = a$	$a * a = a$ $a + a = a$
4.	$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$	$a * (b + c) = (a * b) + (a * c)$
5.	$a \wedge 1 = a$ $a \vee 0 = a$	$a * 1 = a$ $a + 0 = a$
6.	$a \wedge 0 = 0$ $a \vee 1 = 1$	$a * 0 = 0$ $a + 1 = 1$
7.	$\neg(\neg a) = a$	*„minus mal minus ist plus“*
8.	$\neg\,(a \wedge b) = (\neg a) \vee (\neg b)$ $\neg\,(a \vee b) = (\neg a) \wedge (\neg b)$	*De Morganische Gesetze*
9.	$a \wedge \neg a = 0$ $a \vee \neg a = 1$	$1 * 0 = 0$ $1 + 0 = 1$

Weitere Boolesche Operatoren

- Exklusives Oder XOR („entweder oder“)
- Implikation (“Nicht *a* oder *b*”)
- Äquivalenz („genau dann wenn“)

		a ODER b	a UND b	„entweder oder“	Implikation	Äquivalenz
A	B	$A \vee B$	$A \wedge B$	A XOR B	$A \Rightarrow B$	$A \Leftrightarrow B$
0	0	0	0	0	1	1
1	0	1	0	1	0	0
0	1	1	0	1	1	0
1	1	1	1	0	1	1

Beispiele

- `goethe || (geothe && schiller)` = ?
- `schiller && (schiller || goethe)` = ?
- `goethe || (!goethe && schiller)` = ?
- `(goethe && schiller) || (goethe && !schiller)` = ?
- Lösung durch Wahrheitswert-Tabellen

Beispiele – Lösungen

- `goethe || (geothe && schiller) = goethe`
- `schiller && (schiller || goethe) = schiller`
- `geothe || (!goethe && schiller) = goethe || schiller`
- `(goethe && schiller) || (goethe && !schiller) = geothe`

Anwendung in der Suche

- *Werke von Picasso, die keine Gemälde sind und zwischen 1914 und 1916 entstanden sind.*
- *Literarische Werke, deren Titel “Faust” oder “Werther” beinhaltet, jedoch nicht von Goethe sind und nach 1900 erschienen sind.*
- *Szenen in der “Herr der Ringe”-Verfilmung in denen Frodo und Gollum auftreten, nicht aber Sam.*

Reguläre Ausdrücke

Grundlagen Regulärer Ausdrücke

- Reguläre Ausdrücke (engl. *regular expressions* (regex)) sind **abstrakte Ausdrücke** zur beschreiben von Zeichenfolgen
- Ein Regex beschreibt ein Muster (**Pattern**), auf das eine Zeichenfolge passen kann, oder auch nicht (**Matching**)
- Regex können daher für eine verbesserte Suche eingesetzt werden, z.B.:
  - `Herrn? M[ae][iy]er` findet *Herr Meier, Herrn Meier, Herr Maier,...*
  - `\w+\.\w+@uni-jena.de` findet alle FSU-Email-Adressen
  - `\d{4,6}\.\d{4,8}` findet Telefonnummern

Allgemeines zur Syntax

- Die meisten Zeichen ändern ihre Bedeutung nicht, stehen also für sich selber. Darunter
  - Buchstaben (`a, b, c, ...`)
  - Ziffern (`0, 1, 2,...,9`)
  - Das Leerzeichen ␣
- Andere Zeichen kommen in Regex Sonderbedeutungen zu, insbesondere Klammern und manche Satzzeichen. Diesen muss ein Backslash vorangestellt werden, um trotzdem ihre *normale* Bedeutung darzustellen. Z.B:
  - `\.`
  - `\(`
  - `\?`
  - `\\`

Disjunktion und Negation

- Darstellung unterschiedlicher möglicher Zeichen oder Zeichengruppen (**Disjunktion**)
  - Einzelne Zeichen mit eckigen Klammern: `M[ae][iy]er`
  - Eine Spanne (**Range**) von Zeichen mit eckigen Klammer und Minus: `[0-9], [A-Za-z]`
  - Disjunktion beliebiger Zeichenfolgen mit dem Pipe-Operator: `(Herrn|Frau)`
- **Negation** von Zeichen durch Zirkumflex und eckigen Klammern
  - `[^a]`, jedes beliebige Symbol außer “a”
  - `[^0-9]`, keine Zahlen

Zeichenklassen

- `\d`, Ziffern, entspricht `[0-9]`
- `\D`, keine Ziffern
- `\w`, alphanumierische Zeichen (`[a-zA-Z0-9]`)
- `\W`, nicht-alphanumerische Zeichen
- `\s`, Whitespace-Zeichen (Space `\r`, Tab `\t`, Newline `\n`)
- `\S`, Nicht-Whitespace-Zeichen (Zahlen, Buchstaben, Satzzeichen)
- `\b`, Wortgrenzen-Zeichen (boundary), z.B. Whitespace und Interpunktion

Quantoren

- Quantoren (Quantifier) geben an, wie häufig das *vorausgehende* Zeichen wiederholt werden darf
- `?`, null oder ein Mal
- `+`, 1 bis *n*
- `*` , 0 bis *n*
- `{n}`, genau *n* Mal
- `{n,m}`, zwischen und *n* und *m* Mal
- Darüber hinaus gibt viele weitere, insbesondere für UTF-8

Wildcard

- Der Punkt `.` dient als *Wildcard* und matcht jedes beliebige Zeichen.
- Z.B. `w.rf` matcht “wirf” und “warf”, aber auch “wurf”, “werf”, “w4rf”,...
- Gerade die Kombination von Wildcard und Quantifiern (v.a. `+` und `*`) kann zu unerwarteten Effekten führen:
  - Regex: `dies.*\b`
  - *soll* matchen: dieser, dieses, dies,...
  - Text: *Sie findet diesen Sommer besonders schön.*
  - Matcht: *diesen Sommer besonders schön.*

Greediness

- Üblicherweise matchen Regex mit Quantoren die *längsmögliche* Zeichenfolge (**greedy**). Z.B.:
  - Regex: `a+h`
  - Text: *Aaaaaaah!*
  - Match: *aaaaaah* (und nicht *ah*)!
- Das Fragezeichen hinter dem Quantifer ändert dessen Verhalten, so dass jetzt die *kürzestmögliche* Zeichenfolge gematcht wird (**non-greedy, lazy**).
- Beispiel `dies.*?\b`

Escaping

- Um Zeichen mit Sonderbedeutungen darzustellen müssen diese **escaped** werden (Backslash voranstellen)
  - `\(\)`
  - `\{\}`
  - `\[\]`
  - `\.`
  - `\+`
  - `\?`
  - `\|`
  - `\\`

Übung

Auf welche Art von Zeichenkette passen jeweils die folgenden Regex?

- `diese?`
- `g\wb`
- `(1\d|20)\d{2}`
- `\w+\.\w+@\w+\.[\w\-]{2,3}`

Anwendung in der Korpusabfrage

Auf welche Art von Zeichenkette passen jeweils die folgenden Regex?

- Beispiel: Faust 1 `https://www.gutenberg.org/ebooks/2229`
- Fausts Geliebte Margarete wird mit vielen unterschieldichen Namensvarianten bezeichnet (Gretchen, Gretel, Gretelchen,...). Wie finden wir alle Erwähnungen von ihr?
- `(Gret\w*|MARGARETE|Margarete)`
- Technische Umsetzung:
  - Kommandozeilen-Werkzeuge:
    `egrep '(Gret\w*|MARGARETE|Margarete)' faust.txt`

    `http://(www.cs.columbia.edu/~tal/3261/fall07/handout/egrep_mini-tutorial.htm`
  - Anwendungen mit grafischer Oberfläche: `AntConc`

    `http://www.laurenceanthony.net/software.html`

Aufgaben zu nächster Woche

Formalia

- Abgabe bis Montag, den 29.5., 23:59 Uhr
- Per Email an `sven.buechel@uni-jena.de`
- Im PDF-Format

Aufgabe 1: Aussagenlogik

Geben Sie den Wahrheitswertverlauf der folgenden aussagenlogischen Formeln an.

1. $\neg a \vee (a \wedge b)$
2. $(a \wedge b) \vee c$