Folgend: Interessante NLP-Packages

Montag, 30. März 2020 18:15

Praktische Beispiele sind immer in den einzelnen Abschnitten:

koRpus: Bekommen von Statistiken zu Wordcounts, etc. Readability, lexical.diversity

wordcloud: Visualisieren von Wörtern Anhand ihrer Häufigkeit

hunspell - High Performance Stemmer: Checkt Rechtschreibung, parsed text, schlägt richtige Wörter vor, sucht Wortstamm von Wörtern

koRpus

Dienstag, 3. März 2020 14:39

koRpus (https://cran.r-project.org/web/packages/koRpus/index.html)

- Für Infos zum richtigen installieren: https://reaktanz.de/?c=hacking&s=koRpus
- Doku: https://reaktanz.de/R/pckg/koRpus/koRpus.pdf
- Benötigt für vieles: TreeTagger - https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
- Problem: Schaut recht kompliziert aus, hinsichtlich des Formates, was die Daten haben müssen
  - Kann sehr aufwendig sein, bis wir uns da reingearbeitet haben, wie das richtig funktioniert!
- Vorteil: die Readability-Packages sehen sehr mächtig aus - siehe z.B. Doku Seite 45
- Interessante Funktionen
  - lexical diversity-Funktionen: wie oft werden Wörter wiederholt, etc.
    -> für bessere Beschreibung: https://textinspector.com/help/lexical-diversity/
    -> gibt einen Haufen Indexe, da müssten wir uns noch näher reinlesen!
    - $lex.div$

```
C characteristics:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.8255  0.8733  0.8857  0.8952  0.9279  1.0000
   SD
 0.0404


Guiraud's R
   R: 5.2

R characteristics:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.236   5.138   5.391   5.127   5.683   6.155
   SD
 0.9539


Carroll's CTTR
   CTTR: 3.68

CTTR characteristics:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1.581   3.633   3.812   3.625   4.019   4.352
   SD
 0.6745


Uber Index
```

  - Readability-Funktionen z.B. Komplexität der Sätze/Wörter, Reading Ease, etc.
    -> müssten wir uns auch reinlesen
    - $readability$

	index	flavour	raw	grade	age
1	ARI			10.7	
2	Coleman-Liau		72	3.29	
3	Danielson-Bryan DB1		7.29		
4	Danielson-Bryan DB2		52.13	7-8	
5	Dickes-Steiwer		63.19		
6	ELF		2.8		
7	Farr-Jenkins-Paterson		81.8	6	
8	Flesch	en (Flesch)	81.67	6	
9	Flesch-Kincaid			9.87	14.9
10	FOG			13.13	
11	FORCAST			6.3	11.3
12	Fucks		109.6	10.47	
13	Linsear-Write			16.3	
14	LIX		38.41	6	
15	nWS1			2.45	
16	nWS2			3.61	
17	nWS3			5.2	
18	nWS4			7.03	
19	RIX		2	6	
20	SMOG			5.68	10.7
21	Strain		10.56		
22	TRI		2.25		
23	Tuldava		3.8		
24	Wheeler-Smith		28	4	

- $flesch$
- $flesch - kincaid$
- $forcast$

- $guess\_lang$ -> Funktion, was die Sprache erkennt
- $freq.analysis$

	freq
sentences	5.000000
avg.sentence.length	32.200000
words	161.000000
avg.word.length	3.403727
all.characters	762.000000
letters	548.000000
lemmata	1.000000
questions	5.000000
exclamations	0.000000
semicolon	0.000000
colon	0.000000

- $textFeatures$

```
> textFeatures(test2)
  uniqWd    complx sntCt sntLen   syllCt charCt lttrCt      FOG   flesch
1     69 0.4099379     5   32.2 1.093168    762    548 13.12845 81.67001
```

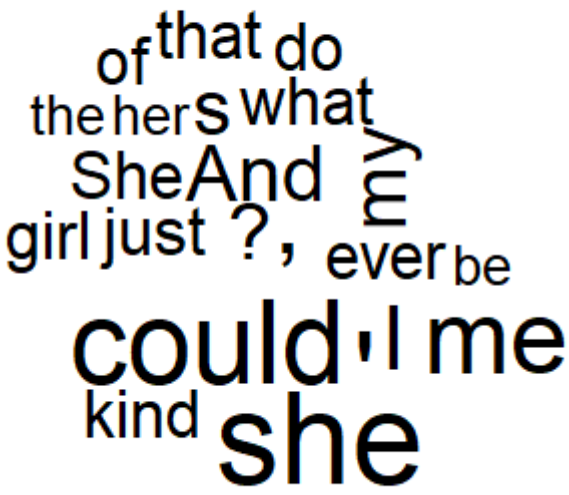
- $treeTag$ - *damit werden Daten vorbereitet*

wordcloud

Montag, 30. März 2020 18:14

Interessante Funktionen:

- *wordcloud* - visualisiert Wörter Anhand ihrer Häufigkeit



hunspell - High Performance Stemmer

Montag, 30. März 2020 18:44

Interessante Funktionen:

- *hunspell*(*text*) - testet auf Rechtschreibfehler im Englischen und gibt Fehler zurück

```
> test$song[1]
[1] "Ahe's My Kind Of Girl"
> hunspell(test$song[1], format = "text")
[[1]]
[1] "Ahe's"
```

- *hunspell_parse*(*text*) - entfernt Satzzeichen

```
> hunspell_parse(test$text[1])
[[1]]
  [1] "Look"      "at"        "her"       "face"      "it's"      "a"         "wonderful" "face"      "And"       "it"        "means"     "something"
 [13] "special"   "to"        "me"        "Look"      "at"        "the"       "way"       "that"      "she"       "smiles"    "when"      "she"
 [25] "sees"      "me"        "How"       "lucky"     "can"       "one"       "fellow"    "be"        "She's"     "just"      "my"        "kind"
 [37] "of"        "girl"      "she"       "makes"     "me"        "feel"      "fine"      "who"       "could"     "ever"      "believe"   "that"
 [49] "she"       "could"     "be"        "mine"      "She's"     "just"      "my"        "kind"      "of"        "girl"      "without"   "her"
 [61] "I'm"       "blue"      "And"       "if"        "she"       "ever"      "leaves"    "me"        "what"      "could"     "I"         "do"
 [73] "what"      "could"     "I"         "do"        "And"       "when"      "we"        "go"        "for"       "a"         "walk"      "in"
 [85] "the"       "park"      "And"       "she"       "holds"     "me"        "and"       "squeezes"  "my"        "hand"      "We'll"     "go"
 [97] "on"        "walking"   "for"       "hours"     "and"       "talking"   "About"     "all"       "the"       "things"    "that"      "we"
[109] "plan"      "She's"     "just"      "my"        "kind"      "of"        "girl"      "she"       "makes"     "me"        "feel"      "fine"
[121] "who"       "could"     "ever"      "believe"   "that"      "she"       "could"     "be"        "mine"      "She's"     "just"      "my"
[133] "kind"      "of"        "girl"      "without"   "her"       "I'm"       "blue"      "And"       "if"        "she"       "ever"      "leaves"
[145] "me"        "what"      "could"     "I"         "do"        "what"      "could"     "I"         "do"
```

- *hunspell_check* - testet darauf ob einzelne Wörter falsch geschrieben sind

```
> hunspell_parse(test$song[1])
[[1]]
[1] "Ahe's" "My"    "Kind"  "Of"    "Girl"

> hunspell_check(unlist(hunspell_parse(test$song[1])))
[1] FALSE  TRUE  TRUE  TRUE  TRUE
```

- *hunspell_suggest* - schlägt Wörter für falsche Wörter vor

```
> unlist(hunspell_parse(test$song[1]))[!hunspell_check(unlist(hunspell_parse(test$song[1])))]
[1] "Ahe's"
> hunspell_suggest(unlist(hunspell_parse(test$song[1]))[!hunspell_check(unlist(hunspell_parse(test$song[1])))])
[[1]]
 [1] "He's"   "Ashe's" "She's"  "Abe's"  "Ave's"  "Che's"  "A he's" "Age's"  "Ache's" "Are's"  "Ale's"  "Ace's"  "Ape's"  "Aye's"  "Awe's"
```

- *hunspell_stem* - gibt Wortstamm für ein Wort aus, damit es in einer Wortwolke zusammengefasst werden kann (z.B. she und she's zusammenfassen) - funktioniert aber nicht Perfekt, siehe "her" -> "h" beim stem

```
> unlist(hunspell_parse(test$text[1]))
  [1] "Look"      "at"        "her"       "face"      "it's"      "a"         "wonderful" "face"      "And"       "it"        "means"     "something"
 [13] "special"   "to"        "me"        "Look"      "at"        "the"       "way"       "that"      "she"       "smiles"    "when"      "she"
 [25] "sees"      "me"        "How"       "lucky"     "can"       "one"       "fellow"    "be"        "She's"     "just"      "my"        "kind"
 [37] "of"        "girl"      "she"       "makes"     "me"        "feel"      "fine"      "who"       "could"     "ever"      "believe"   "that"
 [49] "she"       "could"     "be"        "mine"      "She's"     "just"      "my"        "kind"      "of"        "girl"      "without"   "her"
 [61] "I'm"       "blue"      "And"       "if"        "she"       "ever"      "leaves"    "me"        "what"      "could"     "I"         "do"
 [73] "what"      "could"     "I"         "do"        "And"       "when"      "we"        "go"        "for"       "a"         "walk"      "in"
 [85] "the"       "park"      "And"       "she"       "holds"     "me"        "and"       "squeezes"  "my"        "hand"      "We'll"     "go"
 [97] "on"        "walking"   "for"       "hours"     "and"       "talking"   "About"     "all"       "the"       "things"    "that"      "we"
[109] "plan"      "She's"     "just"      "my"        "kind"      "of"        "girl"      "she"       "makes"     "me"        "feel"      "fine"
[121] "who"       "could"     "ever"      "believe"   "that"      "she"       "could"     "be"        "mine"      "She's"     "just"      "my"
[133] "kind"      "of"        "girl"      "without"   "her"       "I'm"       "blue"      "And"       "if"        "she"       "ever"      "leaves"
[145] "me"        "what"      "could"     "I"         "do"        "what"      "could"     "I"         "do"
> unlist(hunspell_stem(unlist(hunspell_parse(test$text[1]))))
  [1] "look"      "at"        "h"         "face"      "it"        "a"         "wonderful" "face"      "and"       "it"        "mean"      "something"
 [13] "special"   "to"        "me"        "look"      "at"        "the"       "way"       "that"      "she"       "smile"     "when"      "she"
 [25] "see"       "me"        "how"       "lucky"     "can"       "one"       "fellow"    "be"        "she"       "just"      "my"        "kind"
 [37] "of"        "girl"      "she"       "make"      "me"        "feel"      "fine"      "who"       "who"       "could"     "ever"      "believe"
 [49] "that"      "she"       "could"     "be"        "mine"      "she"       "just"      "my"        "kind"      "of"        "girl"      "without"
 [61] "h"         "I'm"       "blue"      "and"       "if"        "she"       "ever"      "leave"     "me"        "what"      "could"     "i"
 [73] "I"         "do"        "what"      "could"     "i"         "I"         "do"        "and"       "when"      "we"        "go"        "for"
```

tidytext

Montag, 30. März 2020 21:02

Interessante Funktionen:

- $cast\_sparse$ - versteh ich noch nicht, was das bringt

Nicht geeignete Packages

Montag, 30. März 2020 18:39

Package	Wieso nicht geeignet?
tau	Ungeeignete Funktionen
languageR	Zu detailierte Informationen zu Lyrik und nicht für normale Texte
zipfR	Auch zu spezifisch für Wortanalysen, nicht für texte geeignet
Mscstexta4r - R-Client für Microsoft Cognitive Services Text Analytics	Wäre super für Sentiment-Analysen, aber nur 5000 Anfragen wären gratis und wir haben deutlich mehr für alle Lieder
openNLP	Funktionen dienen eher zum summarisen von Sätzen oder zum Tokenizen - bringt uns für unsere Fragestellungen ziemlich garnix OpenNLP selbst dürfte extrem leistungsstark sein, aber die Packages, die bereits existieren haben für uns keine sinnvollen Funktionen