

Hallo Welt!



TyReX (TexErk)

...

Type Recognition of Texts



Ziel:

**Automatische Klassifizierung von
Texten nach ihrer Textart**

Anwendung:

z.B. Kategorisierung der Texte für
Suchmaschinen

z.B. Kategorisierung aller Dokumente
eines Unternehmens (zur schnellen
Auffindung relevanter Informationen)

z.B. Textanalysetools





Korpora/Datensets:

- Projekt Gutenberg
- ZeitOnline ('99-'01)
- unannotiert
- 1261 Texte in 4 Kategorien:

Epic	(222)
Drama	(291)
Poetry	(446)
Report	(302)

Aufbereitung:

- Textnormierung
- Parser/Annotator

Vor- und Nachteile:

- + einfachere Weiterverarbeitung durch Normierung
- + Passende Metadaten/Tags hinzufügen
- externes Annotieren, fehlende nützliche Metadaten
- Viele Datentypen für Parser



Datenanalyse und Normierung:

- **Tags:**

<s></s>

<punct> - “.”

<comma> - “,”

<exclamation> - “!”

<question> - “?”

<suspension> - “...”

<colon> - “:”

<thinking> - “_”

<semicolon> - “;”

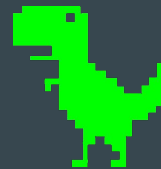
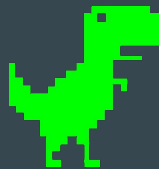
- **Entfernung** von unnötigen chars, Leerzeichen, Zeilenumbrüchen etc.

→ **geordnete** Zeilenumbrüche, Satz- und Zeilengrenzen, etc.

```
1  <s>Abbitte</s>
2
3  <s>Heilig Wesen <exclamation></s><s> gestort hab ich die goldene
4  Gotterruhe dir oft<comma> und der geheimeren<comma>
5  Tiefern Schmerzen des Lebens
6  Hast du manche gelernt von mir<punct></s>
7
8  <s>O vergiß es<comma> vergib <exclamation></s><s> gleich dem Gewolke dort
9  Vor dem friedlichen Mond<comma> geh ich dahin<comma> und du
10 Ruhst und glanzest in deiner
11 Schone wieder<comma> du süßes Licht <exclamation></s>
12
13 <s>Holderlin<comma> Gedichte</s>
```



Struktur und Features



Theorien



Prinzip: Features extrahieren -> an Klassifizierungsalgorithmus übergeben

z.B. nach Zelch/ Engel (2005):

- ‘Lexen’ & Extraktion einzelner Wörter aus Text als Features
- Algorithmus: SVM

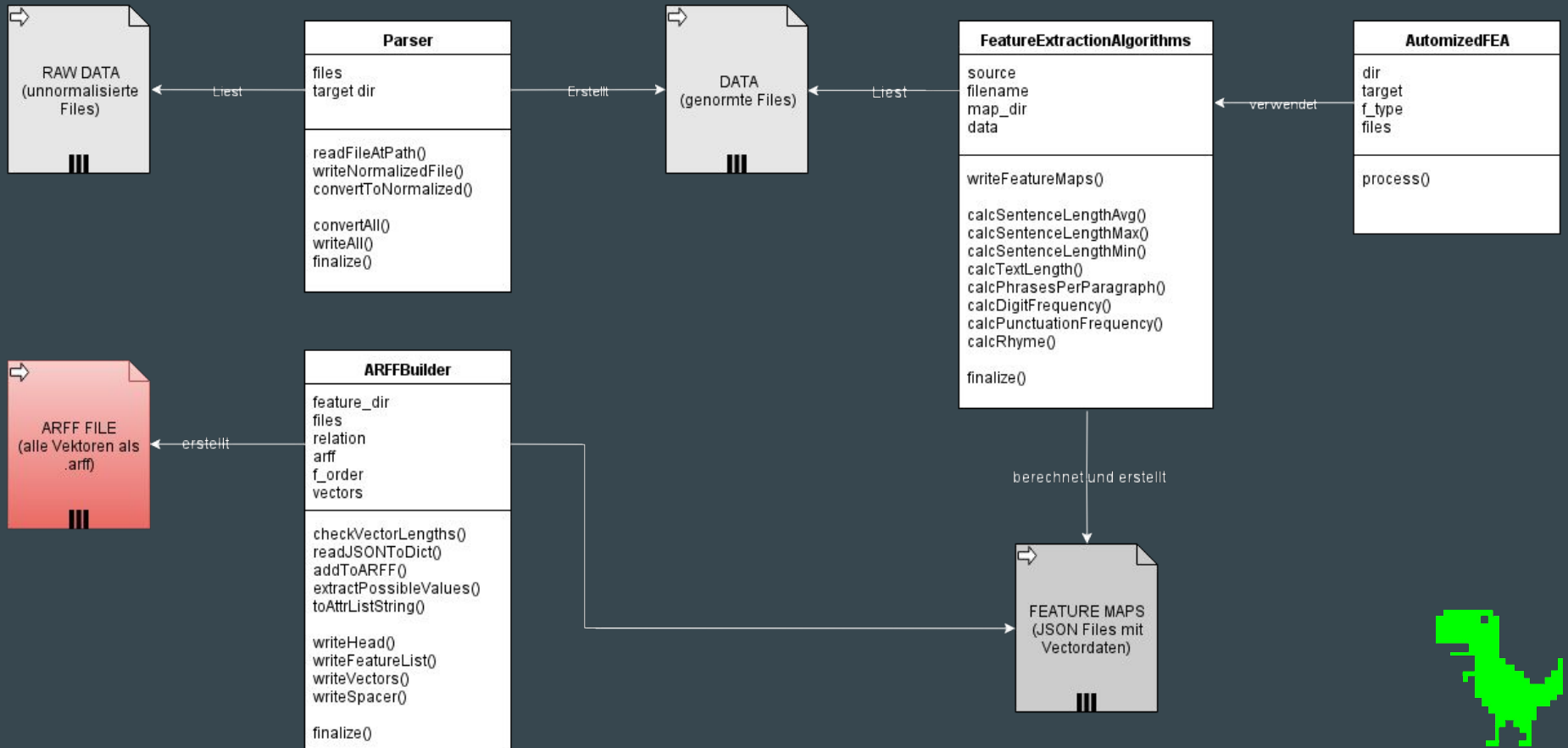
z.B. nach Ghaffari (2015):

- Vektoren aus extrahierten Worten
- Algorithmen: SVM, Naive Bayes, Decision Tree

Tyrex:

- weitere Features, vorerst ohne Wortvektoren (kommt noch)
- Algorithmen mit Weka: Naive Bayes, MultilayerPerceptron, Decision Tree...

Architekturübersicht



Parser

```
phrase_bound = punct + "|" + question + "|" + excl + "|" + "\n{2,}"
phrase_match = "(?=((" + phrase_bound + "|^)((.|\\s)+?)((" + phrase_bound + ")))")"

out = re.sub("[\\*_]|\\#{1,} ", "", out) # remove markdown
out = re.sub("\\[(.*?|\\s*?)\\]|\\|\\|-{2,}|\\t|\\/", "", out) # remove unnecessary characters
out = re.sub("(\\n|^)\\s+\\n", "\\n\\n", out) # remove lines only containing whitespaces
out = re.sub("\\n +", "\\n", out) # remove whitespaces preceding any lines
out = re.sub("^\\s+", "", out) # remove initial whitespaces
out = re.sub(" {2,}", " ", out) # reduce multi space
out = out.replace("\\\\", "\\")

phrases = re.findall(phrase_match, out)
clean_phrases = [phrases[i][2] for i in range(len(phrases)) if phrases[i][3] != phrases[i-1][3]]

out = "".join([phrase[0] + match + phrase[1] for match in clean_phrases]) #sentence bounds
```

Features



aktuell:

- Text Length
- Sentence Length (Min/Max/Average)
- Rhymes
- Digit Frequency
- Punctuation Frequency
- Sentence Per Line Frequency
- Average Word Length
- Word Variance

weitere:

- NE Frequency
- Terminological Congruence
- Verb Occurence
- Direct Speech Frequency
- Mean Sentence Complexity

Text Length:

z.B. poem vs. report vs. epic

- löscht XML-Tags
- zählt die Anzahl der Sätze

```
text = re.sub("<.*?>", "", self.  
source)  
return len(text.split())
```

Sentence Length (Min/Max/Average):

z.B. drama vs. epic

- zählt die Worte pro Satz (ohne Satz-Tags)

```
sentences = [group[0] for group in re.findall("<s>((.|\\s)*?)</s>", self.  
source)]  
return max(map(len, [i.split(" ") for i in sentences]))
```



Digit Frequency:

z.B. report (scientific)

- zählt alle vorkommenden Zahlen im Text

```
nums = len(re.findall("\d+", self.source))  
return float(nums)/len(self.source.split(" "))
```



Punctuation Frequency:

- zählt die Satzzeichen und die Tokens (ohne Satzzeichen)

```
puncts = len(re.findall('<punct>|<exclamation>|<question>|<colon>|<semicolon>|<suspension>|<comma>|<thinking>', self.  
source))  
text_length = len(re.sub('<punct>|<exclamation>|<question>|<colon>|<semicolon>| <suspension>| <comma>|<thinking>', "i",  
self.source))  
return float(puncts)/text_length
```

Sentence Per Line Frequency:

z.B. poem vs. epic/report

- teilt Dokument-Zeilen (Paragraphen) auf -> Elemente einer Liste
- entfernt Leerzeilen
- zählt Satzanfänge (<s>)

```
splitfile = self.source.splitlines()
while '' in splitfile:
    splitfile.remove('')
count = 0
for line in splitfile:
    count += len(re.findall('<s>', line))
return float(count)/len(splitfile)
```



Average Word Length:

z.B. report

- löscht Tags aus Text
- zählt alle Buchstaben

```
clean_text = re.sub('<.*?>', '', self.source))
char = 0
for word in clean_text.split():
    char += len(word)
return float(char)/len(clean_text)
```



Word Variance:

- löscht Tags aus Text
- zählt verschiedene Tokens

```
clean_text = re.sub('<.*?>', '', self.source))
return len(set(clean_text.split()))/len(clean_text.split())
```

zukünftige Features:



NE Frequency:

- z.B. report vs. poem
 - soll Named Entities erkennen und zählen

Terminological Congruence:

- z.B. epic vs. report (news) vs. report (scientific)
 - soll Fachterminologien mit vorhandenen Begriffen vergleichen (lemmatisiert, ohne Stoppwörter)

Verb Occurrence:

- z.B. report (scientific) (= Nominalstil) vs. epic (= Verbalstil)
 - soll Anzahl der Verben zählen

Rhyme Average/ Schemes:

- z.B. poem vs. report (scientific)
 - soll Durchschnitt der sich wiederholenden Endungen zählen/ soll Endungen mit Rheimschema vergleichen

zukünftige Features:



Direct Speech Frequency:

z.B. drama vs all

- braucht Parser der zunächst in der Lage ist Direkte Rede zu erkennen (anhand von “” oder “Sprecher: ...”)

Mean Sentence Complexity:

z.B. epic vs poetry vs drama

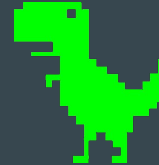
- soll Verschachtelung von Sätzen messen
- z.B. mit Dependenzbäumen
- oder stumpf mit comma frequency

ARFF - Das Ergebnis

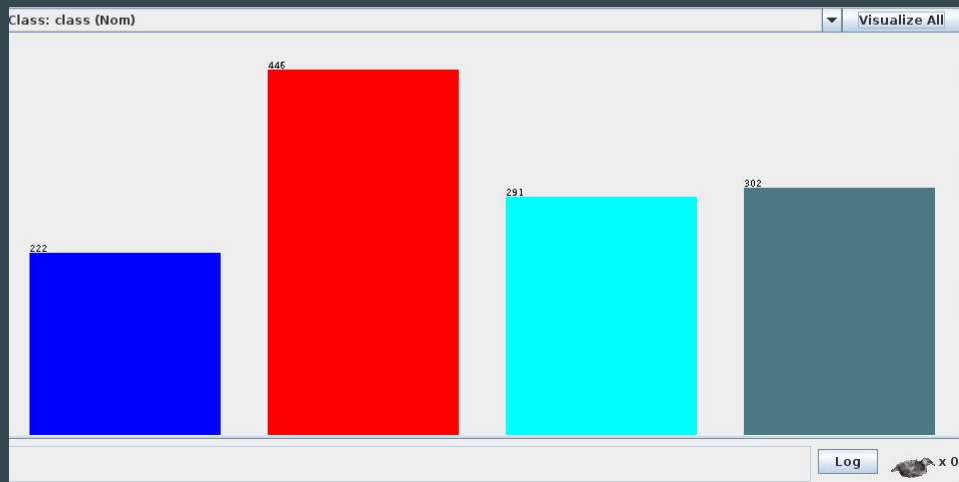
@DATA

0.46353042727665084, 0.0004677268475210477, 0.6939890710382514, 0.03014801173180547, 17.826771653543307, 89, 3, 2317, 0.8260315078769692, poetry
0.873015873015873, 0.0, 1.1666666666666667, 0.022988505747126436, 9.142857142857142, 18, 1, 63, 0.8297872340425532, epic
0.5456730769230769, 0.0, 1.5223880597014925, 0.032768675367953345, 4.4411764705882355, 9, 1, 416, 0.8248465149873601, drama
0.8018867924528302, 0.0, 1.7222222222222223, 0.03402854006586169, 3.967741935483871, 17, 1, 106, 0.8174962292609351, drama
0.8170731707317073, 0.039473684210526314, 2.2857142857142856, 0.03827751196172249, 5.6875, 18, 1, 82, 0.835030549898167, epic
0.73, 0.03225806451612903, 1.75, 0.03201219512195122, 7.571428571428571, 26, 1, 100, 0.8156424581005587, epic
0.735981308411215, 0.0, 0.4594594594594595, 0.02661064425770308, 24.0, 102, 4, 428, 0.8395340097707629, poetry
0.5107296137339056, 0.012987012987012988, 0.6415094339622641, 0.019772071948372924, 13.441176470588236, 33, 1, 932, 0.8539696833258292, report
0.9076923076923077, 0.0, 0.6, 0.027989821882951654, 21.0, 29, 15, 65, 0.8227146814404432, poetry
0.4827744602664217, 0.007157464212678937, 0.512396694214876, 0.01490084985835694, 16.766129032258064, 50, 1, 2177, 0.8647617894545675, report
0.5503355704697986, 0.0, 0.3157894736842105, 0.028659160696008188, 25.166666666666668, 30, 1, 149, 0.814773980154355, poetry
0.8169014084507042, 0.0, 1.2857142857142858, 0.02702702702702703, 8.111111111111111, 14, 1, 71, 0.8197530864197531, epic
0.8382352941176471, 0.0, 0.6666666666666666, 0.03532008830022075, 16.5, 21, 14, 68, 0.8361858190709046, poetry
0.4236295902075572, 0.062085593731163354, 0.7507163323782235, 0.017259450772133325, 7.32824427480916, 40, 1, 1879, 0.8237121831561733, report
0.7107438016528925, 0.0, 0.8421052631578947, 0.02511566424322538, 14.9375, 25, 2, 242, 0.8231841526045488, poetry
0.5185185185185185, 0.0, 0.35294117647058826, 0.015373614587057561, 26.0, 59, 3, 459, 0.8070776255707762, poetry
0.5594855305466238, 0.0, 1.8958333333333333, 0.04001482030381623, 3.89010989010989, 9, 1, 311, 0.8172978505629478, drama
0.3070086338242763, 0.0, 0.6909620991253644, 0.02523778847246959, 16.17721518987342, 95, 1, 3938, 0.8341090425531915, epic

Experimente & Evaluation



Datenverteilung



1261 Texte
verteilt auf 4 Klassen

epic → 222
poetry → 446
drama → 291
report → 302



Baseline - ZeroR (majority voting)

=== Stratified cross- validation ===

=== Summary ===

Correctly Classified Instances	446	35,368 %
Incorrectly Classified Instances	815	64,6312 %
Kappa statistic	0,3667	
Root mean squared error	0,4282	
Relative absolute error	100	%
Root relative squared error	100	%
Total Number of Instances	1261	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0,495	epic
	1	1	0,354	1	0,523	0,496	poetry
	0	0	0	0	0	0,498	drama
	0	0	0	0	0	0,496	report
Weighted Avg.	0,354	0,354	0,125	0,354	0,185	0,496	

=== Confusion Matrix ===

a	b	c	d	← classified as
0	222	0	0	a = epic
0	446	0	0	b = poetry
0	291	0	0	c = drama
0	302	0	0	d = report



Algorithmen

Dataset	(1) function	(2) funct	(3) bayes	(4) meta.	(5) bayes	(6) funct	(7) meta.
tyrex	(20) 87.87	87.31	81.88 *	83.63 *	83.07 *	85.92	56.54 *
	(v/ /*)	(0/1/0)	(0/0/1)	(0/0/1)	(0/0/1)	(0/1/0)	(0/0/1)

Key:

- (1) functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a' -5990607817048210779
- (2) functions.MultilayerPerceptron '-L 0.7 -M 0.2 -N 1000 -V 0 -S 0 -E 20 -H a' -5990607817048210779
- (3) bayes.NaiveBayes '' 5995231201785697655
- (4) meta.MultiClassClassifier '-M 0 -R 2.0 -S 1 -W functions.Logistic -- -R 1.0E-8 -M -1' -3879602011542849141
- (5) bayes.BayesNet '-D -Q bayes.net.search.local.K2 -- -P 1 -S BAYES -E bayes.net.estimate.SimpleEstimator -- -A 0.5' 746037443258775954
- (6) functions.SimpleLogistic '-I 0 -M 500 -H 50 -W 0.0' 7397710626304705059
- (7) meta.MultiBoostAB '-C 3 -P 100 -S 1 -I 10 -W trees.DecisionStump' -6681619178187935148



Multilayer Perceptron

+ Großer Anteil richtig klassifiziert

=== Stratified cross- validation ===

=== Summary ===

Correctly Classified Instances	1105	87.6289 %
Incorrectly Classified Instances	156	12.3711 %
Kappa statistic	0.8306	
Mean absolute error	0.0784	
Root mean squared error	0.0784	
Relative absolute error	21.3833 %	
Root relative squared error	51.0108 %	
Total Number of Instances	1261	

YAY!



Multilayer Perceptron

- + ausgeglichener Precision/Recall
- Unsicherheit zwischen epic/poetry

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.653	0.044	0.759	0.653	0.702	0.919	epic
	0.938	0.029	0.907	0.938	0.922	0.977	drama
	0.868	0.094	0.834	0.868	0.851	0.953	poetry
	0.933	0.005	0.984	0.933	0.988	0.999	report
Weighted Avg.	0.876	0.049	0.874	0.876	0.874	0.964	

=== Confusion Matrix ===

a	b	c	d	← classified as
145	14	61	2	a = epic
1	273	16	1	b = drama
44	13	387	2	c = poetry
1	1	0	300	d = report

<--|
| Features um epic/poetry besser zu trennen nötig
<--|

Nice!



Bewertung der Ergebnisse


- + Relativ **gute Klassifizierung** über **verschiedene Algorithmen** hinweg
- + **87 %** der Texte richtig klassifiziert und damit Baseline weit übertroffen
- + Sehr genaue Klassifizierung von “Reports” und “Drama”
- Keine optimale Trennung zwischen “Epic” und “Poetry”
- Zwar gute Ergebnisse, jedoch bei **nur 4** verschiedene Klassen
- Klassen sehr **grob**
- Verschiedene **Qualität von Features**, Nutzen evtl. erst durch Verbesserung ihrer Berechnung größer oder mit mehr Klassen





Probleme und Überlegungen:

Lösungsansätze:

- weitere Features nötig	-> z.B. NE Aufkommen, Terminologien aus Fachlexika
- Lemmatisierung und POS Tagging, Abhängigkeiten	-> externe Annotationsprogramme, z.B. TreeTagger
- Kombination mit anderen Projekten	-> z.B. NE Annotation
- weitere Experimente	oh wie fein!
- Anwendungsoberfläche	
- mehr (feinere) Klassen	-> z.B. scientific, novel, news, ...

Probleme und Überlegungen:

- Daten ungleich auf Klassen verteilt, evtl. zu spärlich

- Viele Texte in Drama/Epik von gleichem Autor oder sogar aus selben Werken, evtl. Verfälschung

Lösungsansätze:

-> mehr Daten, ausgleichen

-> mehr Daten

-> separates Trainingsset





Fragen und Diskussion



- Welche Features könnten die Ergebnisse optimieren?
 - Welche Features sind eurer Meinung nach nicht sinnvoll?
- Welche weiteren Klassen kann man wählen und woher die Texte dazu nehmen?





Klassifikation:

<http://www.kdnuggets.com/2015/01/text-analysis-101-document-classification.html> - *comparing the number of matching terms in doc vectors*

http://www.python-kurs.eu/text_klassifikation_python.php - *bag of words/ naive bayes*

http://wt.hs-augsburg.de/report/2005/Zelch_Christa_Engel_Stephan/Klassifikation.pdf - *automatische Textklassifizierung mit SVM*

Lewis, David D., Naive (Bayes) at Forty: The independence assumption in informal retrieval, Lecture Notes in Computer Science (1998), 1398, Issue: 1398, Publisher: Springer, Pages: 4-15

K. Nigam, A. McCallum, S. Thrun and T. Mitchell, Text classification from labeled and unlabeled documents using EM, Machine Learning 39 (2000) (2/3), pp. 103-134.



Andere:

<http://www.falkwolfschneider.de/kurs10/Textgattungen.pdf> - *lists different text genres*

Textsorten : Differenzierungskriterien aus linguistischer Sicht / Elisabeth Gülich, Wolfgang Raible (Hrsg.). 2. Aufl., Wiesbaden : Akademische Verlagsgesellschaft Athenaion, c1975; (<http://iucat.iu.edu/iuk/1836130>) - *linguistical criteria*