TyReX (Text Type Recognition)

Projektdokumentation 31.03.2016
Autoren: Lydia Hofmann, Svenja Lohse, Tonio Weidler
Betreuer: Éva Mújdricza-Maydt

Inhaltsverzeichnis

1 Einführung

Das Ziel dieses Projektes ist eine automatische Klassifizierung von Texten nach ihrer Textart. Suchmaschinen könnten das zur Kategorisierung und damit besseren Suche vorhandener Dokumente verwenden und auch andere Unternehmen würden von einem internen Kategoriensystem (mit Kategorien wie u.a. Rechnungen, Mitarbeitergespräche, Rezensionen, etc.) profitieren.

Um dieses Ziel zu erreichen, müssen viele Daten gesammelt, aufbereitet und analysiert werden.
Features, die die Eigenschaften der unterschiedlichen Texte beschreiben, spielen eine wichtige Rolle bei der Genre-Klassifizierung.
. . . Satz zu unserem Ergebnis
Weitere Schritte wären u.a. eine Erweiterung der Feature-Liste, größere Trainingsdatenmenge und z.B. eine einfach zu bedienende Webanwendung.

2 Daten

Die Trainingsdaten stammen aus dem “Projekt-Gutenberg”-Korpus, der viele Werke bekannter Autoren bereit stellt, und “Zeit-Online” dient ebenfalls als Quelle.
Mit diesen unannotierten Texten wurden zwei Korpora erstellt.

Der erste Korpus umfasst 1261 Dateien, die wie folgt in 4 grobe Klassen unterteilt wurden:

```
222 Epische Texte
291 Dramen
302 Artikel
446 Gedichte
```

Der zweite Korpus enthält 11950 Dateien, die wie folgt in x feinere Klassen eingeteilt wurden:

```
efwefwe
fwefwfe
wefwff
```

Die Texte werden durch den "TextNormierer" (Parser) aufbereitet, d.h. Satzzeichen werden durch Tags (</>) ersetzt und unnötige Zeichen entfernt, sodass geordnete Zeilen- und Satzgrenzen entstehen. Durch die Normierung ist die Weiterverarbeitung der Daten einfacher und nützliche Metadaten werden durch die Tag-Setzung eingebunden. Ein Nachteil ist allerdings, dass uns externe Metadaten verloren gehen und der Nomierer viele Datentypen zu verarbeiten hat, wodurch eine optimale Normierung teilweise nicht möglich ist.

AUSSCHNITT NORM_TEXT
AUSSCHNITT TAGGED_TEXT

Zusätzlich lassen wir den TreeTagger die Texte bei der Featureberechnung annotieren, um die so entstandenen POS-Tags und die Baumstruktur in Features verwenden zu können.

3 Struktur

Das einfache Prinzip bisheriger Theorien zu diesem Thema lautet, aus Trainingsdaten Features zu extrahieren und sie an einen Klassifizierungsalgorithmus zu übergeben.
Z.B. Zelch und Engel (2005) haben Wort-Features aus ihren Texten extrahiert, Lexeme gebildet, lemmatisiert und diese Features dann mit einem 'SVM'-Algorithmus verarbeitet. 2015 beschrieb Ghaffari ebenfalls Vektoren aus extrahierten Worten, die er mit den 'SVM'-, 'Naive Bayes'- und 'Decision Tree'-Algorithmen zur Textklassifikation verwendet hatte. Unsere Vorgehensweise ist (weitgehend) ohne Wortvektoren, mit mehr trivialen Features. Mit Weka lassen wir u.a. 'Naive Bayes', 'MultilayerPerceptron' und 'Decision Tree' über die Daten laufen.

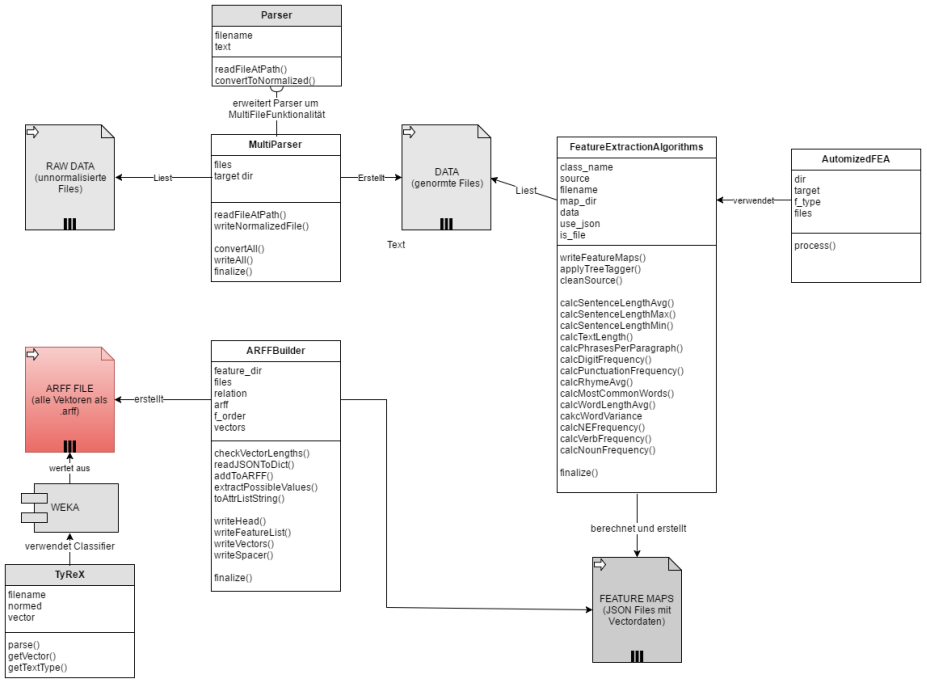

Figure 1: architecture

4 Features

Im Folgenden werden alle bisher verwendeten Features aufgezählt und ihre Funktion grob beschrieben (für einen genauen Einblick kann der Code in "/fea/FeatureExtractionAlgorithms.py" nachvollzogen werden).
- *calcTextLength*
Berechnet die Länge der Texte und ignoriert dabei XML-Tags.
Annahme: z.B. epische Texte sind meist länger als Zeitungsartikel.
- *calcSentenceLengthAvg / calcSentenceLengthMax / calcSentenceLengthMin*
Berechnet die durschnittliche/maximalste/minimalste Anzahl von Wörtern aller Sätze.
Annahme: z.B. während Dramen eher kurze Sätze (u.a. Regieanweisungen) beinhalten, sind epische Werke oder wissenschaftliche Arbeiten eventuell eher langsätzig.
- *calcRhymeAvg*
Zählt alle Aufkommen von Zeilenendungen und berechnet einen Durschnitt der wiederkehrenden Endungen.
Annahme: z.B. sollten Gedichte mehr reimende Endungen enthalten als Zeitungsartikel.
Revision: längere Texte besitzen mehr Endungen, somit eine erhöhte Chance auf gleiche Endungen, und Texte aus der 'Poetry'-Kategorie besitzen weniger reine Reime als gedacht;
Feature muss z.B. mit einer Schema-Prüfung verbessert werden.
- *calcPhrasesPerParagraph*
Berechnet die Zahl der Sätze pro Zeile.
Annahme: Sollte zur besseren Abgrenzung von Gedichten zu anderen Textsorten dienen. Während in Gedichten Sätze häufig über einen gesamten Vers mit mehreren Umbrüchen gehen, tritt bei epischen Texten und Artikeln der erste Umbruch meist erst nach einen gesamten Absatz auf.
Revision: Leider vermindert die Strukturierung der Dateien den Wert des Features. Auch in epischen Texten sind Zeilen künstlich umgebrochen. - *calcDigitFrequency*
Berechnet...
Annahme: z.B....
- *calcPunctuationFrequency*
Berechnet...
Annahme: z.B....
- *calcWordLengthAvg*
Berechnet...
Annahme: z.B....
- *calcWordVariance*
Berechnet, wie unterschiedlich die Wortwahl im Text ist. Es wird die Zahl der einzigartigen Lemmata über die Gesamtzahl an Worten relativiert.
Annahme: In Gedichten ist die Wortwahl häufig abwechslungsreicher, in Dramen und Artikeln vermutlich weniger.

- *calcNEFrequency*
Berechnet...
Annahme: z.B....
- *calcVerbFrequency*
Berechnet...
Annahme: z.B....
- *calcNounFrequency*
Berechnet...
Annahme: z.B....

Diese Features werden durch den FEA berechnet und vom ARFFBuilder in einer ARFF Datei zusammengefasst. Der folgende Ausschnitt zeigt einen Teil dieser ARFF Datei.

```
@relation tyrex


@attribute  NE_frequency    numeric
@attribute  word_variance   numeric
@attribute  digit_frequency numeric
@attribute  noun_frequency  numeric
@attribute  phrases_per_paragraph   numeric
@attribute  punctuation_frequency   numeric
@attribute  rhyme_average   numeric
@attribute  sentence_length_avg numeric
@attribute  sentence_length_max numeric
@attribute  sentence_length_min numeric
@attribute  text_length numeric
@attribute  verb_frequency  numeric
@attribute  word_length_average numeric
@attribute  class   { epic, drama, report, poetry }


@DATA
0.0020035491441982942, 0.35643988018827555, 0.0004677268475210477, 0.023241170072700212, 0.6
0.0, 0.7014925373134329, 0.0, 0.02564102564102564, 1.1666666666666667, 0.022988505747126436,
0.004719101123595505, 0.4606205250596659, 0.0, 0.019325842696629212, 1.5223880597014925, 0.0
0.0148861646234676, 0.7090909090909091, 0.0, 0.021891418563922942, 1.7222222222222223, 0.034
0.015503875968992248, 0.7073170731707317, 0.039473684210526314, 0.020671834625323, 2.2857142
0.006345177664974619, 0.64, 0.03225806451612903, 0.031725888324873094, 1.75, 0.032012195121
0.0023745918670228555, 0.6093023255813953, 0.0, 0.029385574354407838, 0.4594594594594595, 0.
0.00281483294578387, 0.432661717921527, 0.012987012987012988, 0.02753640925223351, 0.641509
0.002178649237472767, 0.7575757575757576, 0.0, 0.026143790849673203, 0.6, 0.027989821882951
```

5 Experimente und Evaluation

Es wurden Experimente auf den grob und fein gegliederten Datensätzen ausgeführt.
Als Baseline wird in beiden Fällen ein ZeroR Algorithmus verwendet der alle Instanzen mit der häufigsten Klasse klassifiziert.
Die Evaluation verwendet CrossValidation mit 10 folds.

Grober Datensatz
Der grob gegliederte Datensatz enthält 1261 Instanzen die auf 4 Klassen verteilt sind. Diese Verteilung verhält sich wie folgt:

```
222 Epische Texte
291 Dramen
302 Artikel
446 Gedichte
```

Die *Baseline* klassifiziert etwa 35% aller Instanzen korrekt. Im folgenden eine detailliertere Übersicht der Ergebnisse der Baseline:

```
Correctly Classified Instances         446               35.3688 %
Incorrectly Classified Instances       815               64.6312 %
Kappa statistic                          0
Mean absolute error                      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
                 0         0          0         0         0          0.498    drama
                 0         0          0         0         0          0.496    report
                 1         1          0.354     1         0.523      0.496    poetry
Weighted Avg.    0.354     0.354      0.125     0.354     0.185      0.496
```

Es wurde ein *Experiment* mit 9 verschiedenen Algorithmen durchgeführt.

```
(1) rules.ZeroR '' 48055541465867954
(2) bayes.NaiveBayes '' 5995231201785697655
(3) functions.Logistic '-R 1.0E-8 -M -1' 3932117032546553727
(4) functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a' -599060781704
(5) functions.SimpleLogistic '-I 0 -M 500 -H 50 -W 0.0' 7397710626304705059
(6) functions.SMO '-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.P
(7) lazy.KStar '-B 20 -M a' 332458330800479083
(8) meta.AdaBoostM1 '-P 100 -S 1 -I 10 -W trees.DecisionStump' -7378107808933117974
```

```
(9) trees.J48 '-C 0.25 -M 2' -217733168393644444
```

Die Ergebnisse zeigen, dass alle gewählten Algorithmen in unterschiedlichem Ausmaß die Baseline übertreffen.

```
Dataset                   (4) function | (1) rules (2) bayes (3) funct (5) funct (6) funct
-------------------------------------------------------------------------------------------
tyrex                     (30)   92.10 |   35.37 *   85.80 *   91.65     92.04     89.56 *
-------------------------------------------------------------------------------------------
                               (v/ /*) |   (0/0/1)   (0/0/1)   (0/1/0)   (0/1/0)   (0/0/1)
```

Die besten Ergebnisse erreicht der MultilayerPerceptron. Logistic, SimpleLogistic und KStar erreichen jedoch Leistungen, die nicht signifikant schlechter sind. Dies ist besonders in Hinsicht auf Logistic und SimpleLogistic von Bedeutung, da ihre Berechnung bedeutend weniger aufwendig ist.

Eine genauere Betrachtung des MultiLayerPerceptrons liefert die folgende Evaluierung:

```
Correctly Classified Instances        1164               92.3077 %
Incorrectly Classified Instances        97                7.6923 %
Kappa statistic                          0.8949
Mean absolute error                      0.0452
Root mean squared error                  0.1816
Relative absolute error                 12.3251 %
Root relative squared error             42.4158 %
Total Number of Instances             1261

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.82      0.03       0.854     0.82      0.837      0.963    epic
                 0.945     0.014      0.952     0.945     0.948      0.981    drama
                 0.993     0.006      0.98      0.993     0.987      0.999    report
                 0.913     0.056      0.898     0.913     0.905      0.973    poetry
Weighted Avg.    0.923     0.03       0.923     0.923     0.923      0.979
```

Bei einer Precision von 92.3077 % sind diese Ergebnisse sehr gut. Die gewählten Features sind offensichtlich ausreichend, um einen sehr genauen Classifier für diese 4 Klassen zu trainieren. Die Klasse report erreicht einen beeindruckenden Recall Wert von 0.993. Quasi alle Zeitungsartikel wurden also auch als solche erkannt. Eventuell ist das aber auch auf ein Overfitting zurückzuführen, basierend auf der über alle Instanzen der Klasse hinweg gleichen Quelle.

Anhand der verschiedenen Precision und Recall Werte für die einzelnen Klassen lässt sich bereits eine Vermutung machen die mit der Confusion Matrix bestätigt wird.

```
=== Confusion Matrix ===
```

```
  a   b   c   d   <-- classified as
182   1   4  35 |   a = epic
  4 275   1  11 |   b = drama
  1   1 300   0 |   c = report
 26  12   1 407 |   d = poetry
```

Während `drama` und `report` sehr gut klassifiziert werden, sowohl hinsichtlich Precision als auch Recall, gibt es Verwirrungen zwischen Epic und Poetry.

Gründe hierfür sind u.a. wohl Ähnlichkeiten in Hinblick auf Zeichensetzung und Schreibstil. Sowohl bezüglich der NounFrequency als auch der VerbFrequency sind Texte beider Klassen kaum zu unterscheiden.
Features, die zur besseren Unterscheidung dieser Klassen dienen sollten, konnten aufgrund der Beschaffenheit der Texte zudem nicht immer richtig greifen. So sind die Epischen Texte leider nicht anhand der Paragraphen umgebrochen. Dadurch kann nicht zwischen Gedichtszeilen und layoutbedingten Umbrüchen in epischen Texten unterschieden werden.

Verbesserte Features (z.B. bzgl. Rhymes) und evtl. Parserfunktionalität, die Paragraphen erkennt, könnten dieses Problem umgeben.

Feiner Datensatz

Der feinere Datensatz enthält insgesamt 11949 erfolgreich geparste Instanzen. Diese verteilen sich folgendermaßen auf insgesamt 11 Klassen

```
lyrik: 4123
dramatik: 105
lustspiel: 134
essay: 20
tragoedie: 482
novelle: 348
maerchen: 1746
sonett: 203
fabel: 1307
roman: 2763
ballade: 128
erzaehlung: 590
```

Ein offensichtliches Problem des Datensatzes ist die ungleiche Verteilung der Daten. Da die Texte per Hand annotiert wurden war eine bessere Annotation in Anbetracht der knappen Zeit nicht möglich.

Als *Baseline* wurde auch hier ZeroR gewählt. Die Baseline erreicht einen Wert von 34.505% korrekt klassifizierten Instanzen.

```
=== Summary ===

Correctly Classified Instances        4123               34.505  %
Incorrectly Classified Instances      7826               65.495  %
```

```
Kappa statistic                          0
Mean absolute error                      0.1315
Root mean squared error                  0.2564
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances            11949

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0         0          0         0         0          0.5      erzaehlung
                 1         1          0.345     1         0.513      0.5      lyrik
                 0         0          0         0         0          0.488    dramatik
                 0         0          0         0         0          0.491    lustspiel
                 0         0          0         0         0          0.5      essay
                 0         0          0         0         0          0.498    tragoedie
                 0         0          0         0         0          0.499    maerchen
                 0         0          0         0         0          0.498    novelle
                 0         0          0         0         0          0.495    sonett
                 0         0          0         0         0          0.499    roman
                 0         0          0         0         0          0.494    ballade
                 0         0          0         0         0          0.499    fabel
Weighted Avg.    0.345     0.345      0.119     0.345     0.177      0.499
```

Insbesondere die durschnittliche Precision von 0.119 sollten bessere Algorithmen übertreffen können.

Bei einem Experiment mit 7 verschiedenen Algorithmen hat sich X als bester Classifier herausgestellt. Es wurden die folgenden Algorithmen verwendet:

```
(1) rules.ZeroR '' 48055541465867954
(2) bayes.NaiveBayes '' 5995231201785697655
(3) functions.SimpleLogistic '-I 0 -M 500 -H 50 -W 0.0' 7397710626304705059
(4) functions.Logistic '-R 1.0E-8 -M -1' 3932117032546553727
(5) functions.SMO '-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.P
(6) trees.J48 '-C 0.25 -M 2' -217733168393644444
(7) functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a' -599060781704
```

Neben dem MultiLayerPerceptron haben sich die restlichen Algorithmen bis auf NaiveBayes als ähnlich präzise herausgestellt. Die Ergebnisse des J48 Baum sogar innerhalb der Signifikanzschwelle. Alle Algorithmen übertreffen die Baseline.

```
Dataset                   (1) rules.Ze | (2) bayes (3) funct (4) funct (5) funct (6) trees (
-------------------------------------------------------------------------------------------
tyrex                      (9)   34.50 |   47.68 v   69.16 v   69.20 v   65.86 v   70.39 v
-------------------------------------------------------------------------------------------
                          (v/ /*) |   (1/0/0)   (1/0/0)   (1/0/0)   (1/0/0)   (1/0/0)   (1/
```

Nimmt man den MultiLayerPerceptron genauer unter die Lupe, ergeben sich die

folgenden Evaluationsergebnisse:

```
=== Summary ===

Correctly Classified Instances        8567               71.6964 %
Incorrectly Classified Instances      3382               28.3036 %
Kappa statistic                          0.6283
Mean absolute error                      0.066
Root mean squared error                  0.187
Relative absolute error                 50.1964 %
Root relative squared error             72.96   %
Total Number of Instances            11949

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.134     0.012      0.374     0.134     0.197      0.829    erzaehlung
                 0.923     0.119      0.803     0.923     0.859      0.952    lyrik
                 0.19      0.003      0.364     0.19      0.25       0.956    dramatik
                 0.142     0.005      0.26      0.142     0.184      0.933    lustspiel
                 0.25      0          1         0.25      0.4        0.676    essay
                 0.61      0.021      0.544     0.61      0.575      0.941    tragoedie
                 0.663     0.052      0.687     0.663     0.674      0.915    maerchen
                 0.023     0.003      0.19      0.023     0.041      0.865    novelle
                 0         0          0         0         0          0.924    sonett
                 0.893     0.118      0.696     0.893     0.782      0.944    roman
                 0.008     0          1         0.008     0.016      0.856    ballade
                 0.543     0.032      0.675     0.543     0.602      0.901    fabel
Weighted Avg.    0.717     0.081      0.677     0.717     0.681      0.928

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l   <-- classified as
   79   19    1   12    0   22   43    8    0  381    0   25 |    a = erzaehlung
    9 3806   10   12    0   40   96    3    0   28    0  119 |    b = lyrik
    1   20   20    4    0   54    0    0    0    5    0    1 |    c = dramatik
    1   18   11   19    0   62    2    0    0   18    0    3 |    d = lustspiel
    2    3    0    1    5    3    0    0    0    3    0    3 |    e = essay
    5  110   11    9    0  294    1    2    0   46    0    4 |    f = tragoedie
   23  164    2    3    0   13 1157    5    0  250    0  129 |    g = maerchen
   16    2    0    0    0    4   38    8    0  273    0    7 |    h = novelle
    1  197    0    0    0    0    2    0    0    0    0    3 |    i = sonett
   66   36    0    4    0   33  113   12    0 2468    0   31 |    j = roman
    0   93    0    0    0    1   15    0    0    1    1   17 |    k = ballade
    8  269    0    9    0   14  218    4    0   75    0  710 |    l = fabel
```

Obwohl die erreichten Werte in Precision, Recall und F-Measure relativ hoch sind,

zeigt ein Blick auf die detailliertere Auswertung, dass diese Evaluierungsmaße nur in den Klassen gute Werte erreichen, für die viele Instanzen verfügbar sind. Precision Werte über 0.6 erreichen lediglich die 4 größten Klassen (lyrik, märchen, roman, fabel). Einen Recall Wert über 0.6 erreichen lediglich die Klassen Lyrik, Tragödie, Märchen und Roman. Auffällig sind besonders die hohen Recall Werte von ~ 0.9 der Klassen lyrik und roman.

Zurückzuführen sind diese Beobachtungen auf zum einen die ungleiche Verteilung der Klassen, die ein gutes trainieren des Models erschwert. Es kann angenommen werden dass die Modelle für kleinere Klassen stark overfitten.
Zudem ist ersichtlich dass die gewählten Features allein nicht ausreichen, um eine so feine Unterteilung vorzunehmen. Selbst für einen Menschen kann eine derartige Unterteilung schwer sein, deshalb ist dies ein komplexeres Problem.

6 Auswertung

blablabla

7 Aussichten

Eine Verbesserung der Klassifizierung könnte weiterhin mit größeren und ausgewogeneren Datenmengen erzielt werden. Diese sind allerdings meist schwierig zu finden, vorallem sobald eine Aufteilung in feinere Klassen erfolgen soll.
Bei der Anwendung der Feature-Methoden fällt auf, dass einige Ergebnisse nicht immer wie erwartet ausfallen:
u.a. der durchschnittliche Reimwert bei 'Poetry' ist vergleichsweise sehr niedrig, obwohl dieser Feature eigens für Gedichterkennung gedacht war. Eine Entwicklung weiterer Features (die z.B. Terminologien vergleichen) ist ratsam, ebenfalls könnte man durch weitere Experimente und weitere feinere Klasse ein besseres Ergebnis erzielen. Kombinationen dieses Projekts mit anderen Forschungsprojekten wären eine Überlegung wert.

8 Literatur

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*