Prognose der erfolgreichen Finanzierung von Kickstarter-Projekten auf Basis linguistischer Merkmale

Martin Ratzmann

2025-04-10

Einleitung:

Die Interaktionale Linguistik baut auf der Grundannahme auf, dass sprachliche Strukturen (z. B. Satzbau, Wortwahl) nicht nur abstrakte Systeme sind, sondern aktiv hervorgebracht, geformt und im sprachlichen Kontext gebraucht wird. Sie verbindet Methoden der Gesprächsanalyse, Pragmatik und Grammatikforschung und betont die Bedeutung von Kontext, non-verbalen Faktoren und gemeinsamer Bedeutungsherstellung. Im Mittelpunkt stehen dabei konkrete Kommunikationssituationen und die dynamische Wechselwirkung zwischen Sender und Empfänger. Als zentraler Prozess für die korpuslinguistische als auch die interaktionslinguistischen Analyse wird die Kodierung hevorgehoben (Proske, 2024). “Kodieren bedeutet, Daten systematisch und nach eindeutigen, dokumentierten Kriterien zu kategorisieren” (Proske, 2024; S.175)

What or Why?

Zwei vorliegende Analysen untersuchen die Erfolgswirksamkeit von textuallen Ansprachen in Projekten, wie sie auf “Kickstarter” vorliegen, sowie die mögliche Übertragbarkeit relevanter inhaltlicher oder pragmatischer Faktoren im resultierenden Sprachmodell auf neue Texte.

Eine inhaltsanalitische Betrachtung zeigt signifikante Beziehungen zwischen spezifischem Inhalt und Erfolg (Bouncken & Kai, 2024) - der resultierende Effekt ist jedoch sehr klein. Möglicherweise liegt dies darin begründet, dass die Vielfalt unterschiedlicher Ideen nicht gut erklärt werden kann, wenn sie auf die Verwendung dergleichen Inhalte reduziert werden. Ein völlig anderer Ansatz untersucht die Betrachtung von Wortformen, also der Ebene der Wortsyntax. Dabei werden Ähnlichkeiten und Unterschiede nicht auf der inhaltlichen Ebene untersucht, sondern im Verhältnis von Wortarten (Verben, Adjektive, Substantivierung etc.) zueinander.

In der “Computational Linguistics” (Sprachverarbeitung) werden Kategorien als sogenannte “Universal Part-of-Speech tags” (UPOS) eingesetzt, um spezfische Wortarten standardisiert zu beschreiben.

UPOS-Tag	Beschreibung	Beispiele
ADJ	Adjektiv	schön, groß, wichtig
ADP	Adposition (Präposition/Postposition)	auf, unter, bei
ADV	Adverb	schnell, gestern, sehr
AUX	Hilfsverb	sein, haben, werden
CCONJ	Koordinierende Konjunktion	und, oder, aber
DET	Determinierer (Artikel, Pronomen)	der, ein, diese
INTJ	Interjektion	ach, oh, hallo
NOUN	Nomen	Hund, Idee, Tisch
NUM	Numerale	eins, zwei, hundert
PART	Partikel	nicht, zu (in Infinitiven)
PRON	Pronomen	ich, du, er, sie
PROPN	Eigenname	Berlin, Angela, Amazon
PUNCT	Satzzeichen	., ?, !
SCONJ	Subordinierende Konjunktion	weil, obwohl, dass
SYM	Symbol	$, %, +
VERB	Vollverb	gehen, schreiben, arbeiten
X	Sonstiges (unbekannt, fremdsprachig)	(z.B. japanische Zeichen)

Training des Sprachmodell

Als Textdaten wurden die Beschreibung des Vorhabens (story) und der Risiken (risks) für erfolglose und erfolgreiche Projekte im Jahr 2016 verwendet, wobei eine Dummy-Variable für Erfolg (=1) erstellt wurde.

In den Textdaten für 2016 wurden die absoluten Häufigkeiten verschiedener Wortformen nach "Universal-POS-Tags (UPOS) bestimmt und hier geladen und (für erfolglose und erfolgreiche Projekte) zusammengefügt.

```
# Pakete laden
library(dplyr)
library(tidyr)
library(ggplot2)
library(readr)

# Dateien laden (Pfad ggf. anpassen)
success_data <- read.csv("~/Bookdown/EJIS mit Pragmatismus/success_kick_all_upos_counts.csv")
fail_data <- read.csv("~/Bookdown/EJIS mit Pragmatismus/failed_kick_all_upos_counts.csv")

# Beide um eine neue Spalte "Category" ergänzen
success_data$Category <- "Success"
fail_data$Category <- "Fail"

# Zusammenführen
combined_data <- bind_rows(success_data, fail_data)
```

Hier werden die absoluten Häufigkeiten in relative Häufigkeiten transformiert dargestellt

Um die absoluten Häufigkeiten vergleichbar zu machen, werden die absoluten Häufigkeiten ("Freq") als Anteil an "Token" bestimmt (relative Häufigkeit). Die Daten werden ins Wide-Format transformiert.

```
combined_data$Freq = combined_data$Freq*100/combined_data$Tokens

combined_wide <- combined_data %>%
  pivot_wider(names_from=Var2, values_from=Freq)
```

Fehlende Werte ("fehlende Vorkommen von Wortarten") werden hier als Null kodieren.

```
combined_wide[ , 5:21][is.na(combined_wide[ , 5:21])] <- 0
```

Das Datenframe schließt jetzt alle (erfolglose und erfolgreiche) Projekte ein. Eine zusätzliche Dummy-Variable "success" wird für Erfolglos=0 vs. Erfolg=1 erstellt.

Zwei t-Test's sollen aufzeigen, ob es signifikante Unterschiede zwischen erfolglosen und erfolgreichen Projekttexten gibt, die sich im Einsatzes von Adjektiven bzw. Verben aufzeigen lassen.

```
combined_wide$success <- ifelse (combined_wide$Category=="Success",1,NA)
combined_wide$success <- ifelse (combined_wide$Category=="Fail",0,combined_wide$success)

# T-Test für unabhängige Stichproben
t_test_result <- t.test(ADJ ~ success, data= combined_wide)
t_test_result
```

```
##
##  Welch Two Sample t-test
##
## data:  ADJ by success
## t = 7.0771, df = 24454, p-value = 1.511e-12
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  0.1101281 0.1944956
## sample estimates:
## mean in group 0 mean in group 1
##        6.671061        6.518749
```

```
t_test_result <- t.test(VERB ~ success, data= combined_wide)
t_test_result
```

```
##
##  Welch Two Sample t-test
##
## data:  VERB by success
## t = 32.1, df = 24300, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  0.8856196 1.0008075
## sample estimates:
## mean in group 0 mean in group 1
##        11.63888        10.69566
```

Eine logistische Regression soll aufzeigen, ob das relative Vorkommen spezifischer Wortarten für erfolgreiche Projekte typisch ist. Zuerst wird das Nullmodell als Baseline bestimmt, um dann herauszufinden, ob die Prädiktoren (Wortarten) das Modell besser erklären können.

Interjektion (INTJ, z.B. "ach", "oh", "hallo") und sonstige Unbekannte (z.B. fremdsprachige Zeichen) wurden ausgeschlossen.

```r
#hist(combined_wide$VERB + combined_wide$ADJ + combined_wide$NOUN)
#table (combined_wide$success)
model0 <- glm (success ~ 1, family = binomial(), data=combined_wide)
summary(model0)
```

```
##
## Call:
## glm(formula = success ~ 1, family = binomial(), data = combined_wide)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.02199    0.01256  -1.751   0.0799 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 35180  on 25378  degrees of freedom
## Residual deviance: 35180  on 25378  degrees of freedom
## AIC: 35182
##
## Number of Fisher Scoring iterations: 3
```

```r
# Testmodell1
model1 <- glm (success ~ ADJ + ADP + ADV + AUX + CCONJ + DET + NOUN + NUM + PART + PRON + PROPN + PUNCT
summary(model1)
```

```
##
## Call:
## glm(formula = success ~ ADJ + ADP + ADV + AUX + CCONJ + DET +
##     NOUN + NUM + PART + PRON + PROPN + PUNCT + SCONJ + SYM +
##     VERB, family = binomial(), data = combined_wide)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.127995   0.422061  -0.303 0.761690
## ADJ          0.022837   0.009261   2.466 0.013664 *
## ADP          0.111790   0.008231  13.581  < 2e-16 ***
## ADV          0.013472   0.010640   1.266 0.205477
## AUX         -0.094112   0.010140  -9.281  < 2e-16 ***
## CCONJ       -0.017348   0.012008  -1.445 0.148533
## DET          0.028026   0.008444   3.319 0.000903 ***
## NOUN        -0.067964   0.006825  -9.959  < 2e-16 ***
## NUM          0.061340   0.014253   4.304 1.68e-05 ***
## PART        -0.120566   0.016024  -7.524 5.31e-14 ***
## PRON         0.033429   0.008080   4.137 3.51e-05 ***
## PROPN        0.036021   0.005724   6.293 3.12e-10 ***
## PUNCT        0.101089   0.006482  15.595  < 2e-16 ***
## SCONJ        0.120676   0.020217   5.969 2.38e-09 ***
## SYM          0.202738   0.025422   7.975 1.52e-15 ***
## VERB        -0.083926   0.009830  -8.538  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 35180  on 25378  degrees of freedom
## Residual deviance: 32224  on 25363  degrees of freedom
## AIC: 32256
##
## Number of Fisher Scoring iterations: 4
```

Ein OMNIBUS-Test vergleicht, ob das Modell mit Prädiktoren das Ergebnis (erfolgreich) besser erklärt.

```
model.chi = model1$null.deviance - model1$deviance
df.chi = model1$df.null -model1$df.residual
pval.chi = 1 - pchisq(model.chi, df.chi)
round(pval.chi,3)
```

```
## [1] 0
```

Das Ergebnis der logistischen Regression weist über die Koeffizienten darauf hin, wie stark sich die Log-Odds von Erfolg ändern, wenn der jeweilige Prädiktor (relative Worthäufigkeiten) um eine Einheit steigt (bei konstanten anderen Variablen).

Um diese "Effekte" besser greifbar zu machen, können die ODDs-Ratios berechnet werden. Die Odds Ratio (OR) istdie Exponentialfunktion des Koeffizienten und sagt direkt, um welchen Faktor sich die Chance auf "success" (= 1) verändert, wenn sich der Prädiktor um eine Einheit verändert.

Ein Prädiktor mit einer ODDs Ratio größer 1 zeigt auf, dass die Chancen auf Erfolg steigen, wenn dieser Prädiktor steigt.

Umgekehrt weist eine ODDs Ratio kleiner 1 auf einen Prädiktor, bei durch dessen Steigerung die Chance auf "success" sinkt.

Ist die ODDs Ratio eines Prädiktors gleich 1, hat die Veränderung des Prädiktors keine Auswirkungen auf die Chancen.

```
exp(cbind(OR=coef(model1), confint(model1)))
```

```
## Waiting for profiling to be done...
```

```
##                    OR     2.5 %    97.5 %
## (Intercept) 0.8798575 0.3818561 2.0001927
## ADJ         1.0230994 1.0047115 1.0418574
## ADP         1.1182783 1.1004108 1.1364977
## ADV         1.0135628 0.9926608 1.0349435
## AUX         0.9101804 0.8922539 0.9284371
## CCONJ       0.9828020 0.9599336 1.0061989
## DET         1.0284228 1.0115545 1.0455999
## NOUN        0.9342939 0.9218948 0.9468987
## NUM         1.0632606 1.0339720 1.0933924
## PART        0.8864184 0.8589808 0.9146691
## PRON        1.0339938 1.0178128 1.0505548
## PROPN       1.0366781 1.0251664 1.0484380
## PUNCT       1.1063751 1.0924621 1.1205820
## SCONJ       1.1282594 1.0844513 1.1738959
## SYM         1.2247512 1.1655678 1.2877274
## VERB        0.9194989 0.9019481 0.9373827
```

Steigt der Anteil von Verben (VERB) in der Projektbeschreibung um eins, ist die Wahrscheinlichkeit für die erfolgreiche Finanzierung geringer, d.h. es gibt eine geringere Chance (OR) auf eine erfolgreiche Finanzierung.

Es kann hier auch ein Pseudo-R-square (Cox and Snell) als Gütemaß bestimmt werden, dass die Relevanz des Modells aufzeigt.

```
n = length(model1$residuals)
R2cs = 1-exp((model1$deviance-model1$null.deviance)/n)
R2cs
```

```
## [1] 0.1099285
```

```
R2n = R2cs/(1-exp(-(model1$null.deviance/n)))
R2n
```

```
## [1] 0.1465772
```

```
#write.csv(combined_wide, "~/Bookdown/EJIS mit Pragmatismus/combined_wide.csv")
```

Ein Cox & Snell - R^2 weist nur einen Bereich von 0.00 bis 0.75 auf, somit sind die Resultate nicht vergleichbar mit dem Standard-R^2 der Regression. Vorhersage der Wahrscheinlichkeit für "Success" bei einem Anteil von 10 Prozent Verben.

Das korrigierte R^2 nach Nagelkerke (Skalierung von 0 bis 1) ist entsprechend etwas größer. Das entspricht zudem einem geringen (bis mittleren) Effekt.

Test des Sprachmodells

Prognosen für den Zeitraum von 2017-2023

Die im Jahr 2016 bestimmten ODDs sollen nun als "Sprachmodell" auf die Daten von 2017 übertragen werden, um die richtigen Zuordnungen von erfolglosen und erfolgreichen Projektbeschreibungen zu prüfen.

Auf der Grundlage der im Sprachmodells ermittelten ODDs aus den Trainingsdaten (2016) wird der Erfolg der Projekte von 2017-2023 prognostiziert, indem die Wahrscheinlichkeiten ob Erfolg bestimmt wird.

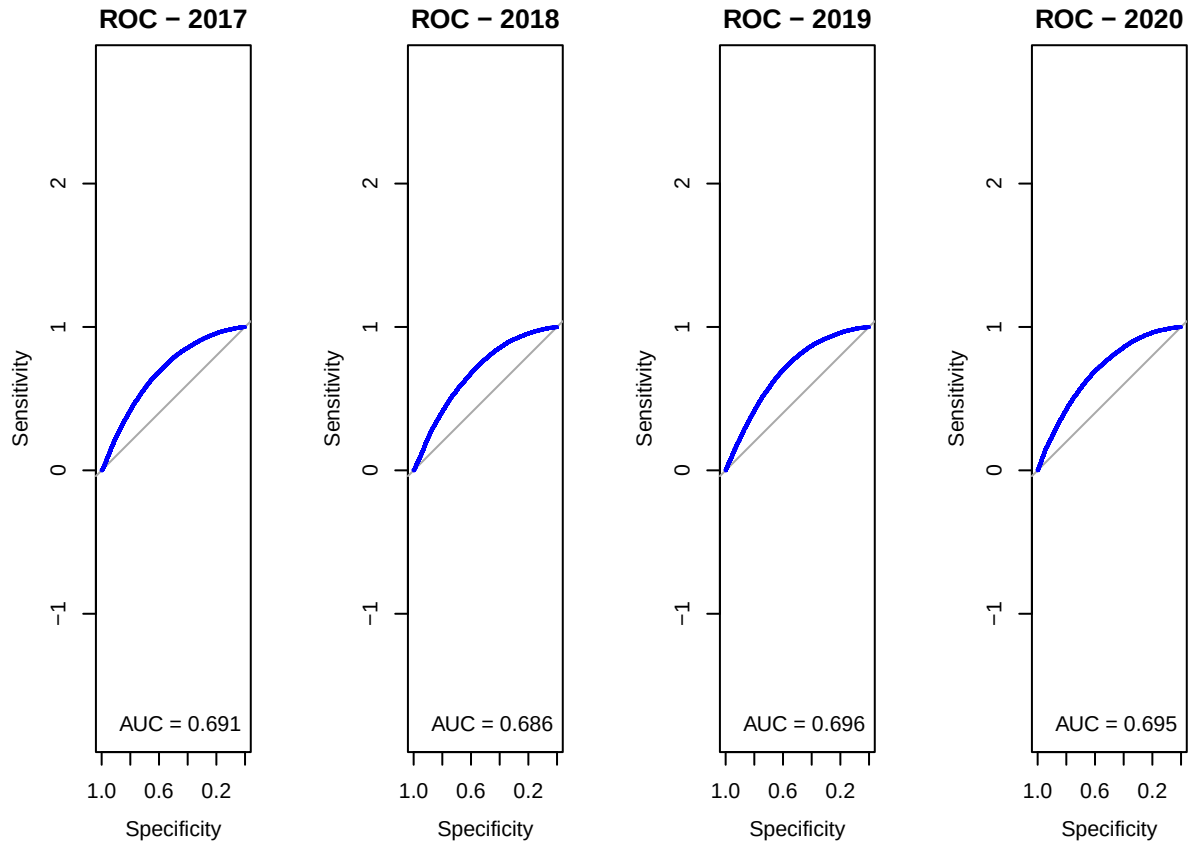
```
# Annahme: POS-Features (kontinuierlich)
# coefs wie oben:
coefs <- log(c(
  Intercept = 0.8798575,
  ADJ = 1.0230994,
  ADP = 1.1182783,
  ADV = 1.0135628,
  AUX = 0.9101804,
  CCONJ = 0.9828020,
  DET = 1.0284228,
  NOUN = 0.9342939,
  NUM = 1.0632606,
  PART = 0.8864184,
  PRON = 1.0339938,
  PROPN = 1.0366781,
  PUNCT = 1.1063751,
  SCONJ = 1.1282594,
  SYM = 1.2247512,
  VERB = 0.9194989
))
```

Hier werden die Test-Textdaten für die Jahre 2017 bis 2023 eingelesen und die Wahrscheinlichkeiten für Erfolg bestimmt. Als kritischer Wert für die Klassifikation als erfolglos vs. erfolgreich wird zunächst der Schwellenwert p=0.5 verwendet. Dieser Schwellenwert wird später auf Grundlage des Youden-Index (J) optimiert.

```
# Pakete laden
library(dplyr)
library(tidyr)
library(ggplot2)
library(readr)

years <- data.frame(year = c("2017", "2018", "2019", "2020", "2021", "2022", "2023" ))
folder <- "~/Markdown/Kickstarter_Ling/data/kickstarter/"

# ein datenframe erstellen, um darin die Gütekriterien zu speichern
quality <- data.frame(
  YEAR = character(),
  TN = numeric(),
  FN = numeric(),
  TP = numeric(),
  FP = numeric(),
  SENS = numeric(),
  SPEC = numeric(),
```

```
  ACCU = numeric(),
  PPV = numeric(),
  NPV = numeric(),
  F1 = numeric(),
  J= numeric()
)

library(pROC)
library(tidyverse)  # für bind_rows, pivot_wider etc.

# Cutoff-Werte definieren
cutoff_values <- seq(0.1, 0.5, by = 0.1)

# Ergebnis-Tabellen vorbereiten
all_quality <- data.frame()
threshold_results <- data.frame()

# ----------------------------------------
# ROC-Kurven pro Jahr EINMAL plotten
# ----------------------------------------

par(mfrow = c(nrow(years)/4,4))  # eine Zeile, je Jahr ein Plot

for (i in 1:nrow(years)) {
  # Daten einlesen
  success_data <- read.csv(paste0(folder, years$year[i], "_success_all_upos_counts.csv"))
  failed_data  <- read.csv(paste0(folder, years$year[i], "_fail_all_upos_counts.csv"))

  success_data$Category <- "Success"
  failed_data$Category  <- "Failed"

  combined_data <- bind_rows(success_data, failed_data)
  combined_data$Freq <- combined_data$Freq * 100 / combined_data$Tokens

  combined_wide <- combined_data %>%
    pivot_wider(names_from = Var2, values_from = Freq)

  combined_wide[, 5:21][is.na(combined_wide[, 5:21])] <- 0
  combined_wide$success <- ifelse(combined_wide$Category == "Success", 1, 0)

  combined_wide$logit <- coefs["Intercept"] +
    coefs["ADJ"]   * combined_wide$ADJ +
    coefs["ADV"]   * combined_wide$ADV +
    coefs["AUX"]   * combined_wide$AUX +
    coefs["CCONJ"] * combined_wide$CCONJ +
    coefs["DET"]   * combined_wide$DET +
    coefs["NOUN"]  * combined_wide$NOUN +
    coefs["NUM"]   * combined_wide$NUM +
    coefs["PART"]  * combined_wide$PART +
    coefs["PRON"]  * combined_wide$PRON +
    coefs["PROPN"] * combined_wide$PROPN +
    coefs["PUNCT"] * combined_wide$PUNCT +
    coefs["SCONJ"] * combined_wide$SCONJ +
```

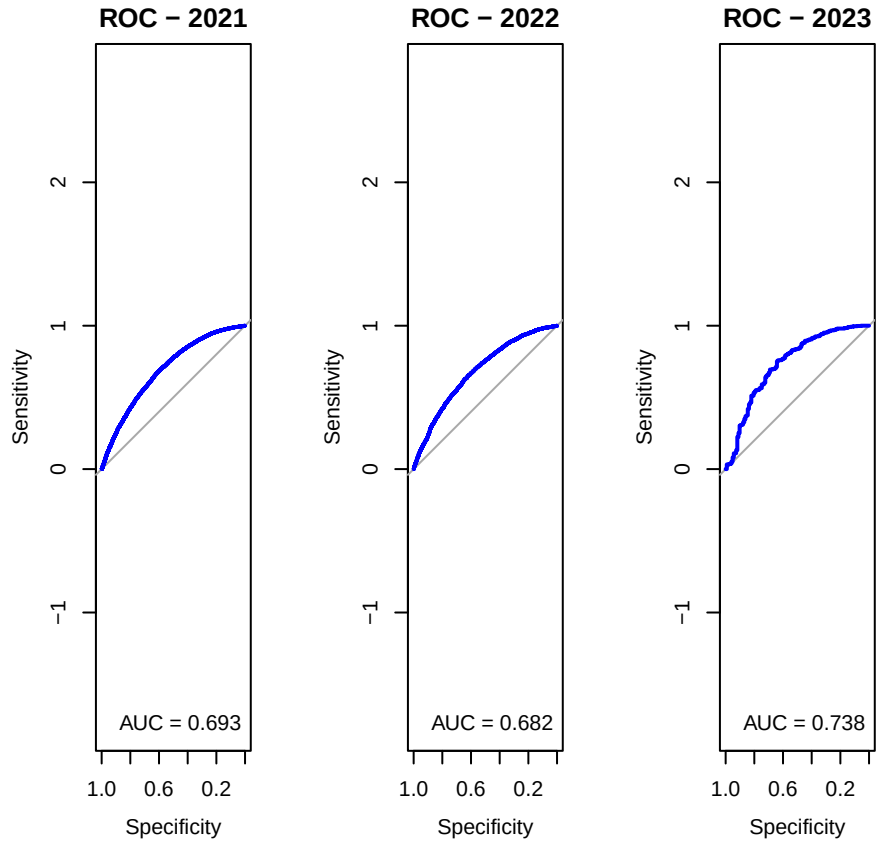
```
    coefs["SYM"]   * combined_wide$SYM +
    coefs["VERB"]  * combined_wide$VERB

  combined_wide$prob <- 1 / (1 + exp(-combined_wide$logit))

  roc_obj <- roc(combined_wide$success, combined_wide$prob)
  plot(roc_obj, main = paste("ROC -", years$year[i]), col = "blue")
  auc_val <- auc(roc_obj)
  legend("bottomright", legend = paste("AUC =", round(auc_val, 3)), bty = "n")
}
```




```
# ---------------------------------------
# Metriken für verschiedene Cutoffs berechnen
# ---------------------------------------
for (cutoff in cutoff_values) {
  prob_cutoff <- cutoff
  quality <- data.frame()

  for (i in 1:nrow(years)) {
    # Daten wie zuvor
    success_data <- read.csv(paste0(folder, years$year[i], "_success_all_upos_counts.csv"))
    failed_data  <- read.csv(paste0(folder, years$year[i], "_fail_all_upos_counts.csv"))

    success_data$Category <- "Success"
    failed_data$Category  <- "Failed"
```

```r
combined_data <- bind_rows(success_data, failed_data)
combined_data$Freq <- combined_data$Freq * 100 / combined_data$Tokens

combined_wide <- combined_data %>%
  pivot_wider(names_from = Var2, values_from = Freq)

combined_wide[, 5:21][is.na(combined_wide[, 5:21])] <- 0
combined_wide$success <- ifelse(combined_wide$Category == "Success", 1, 0)

combined_wide$logit <- coefs["Intercept"] +
  coefs["ADJ"]   * combined_wide$ADJ +
  coefs["ADV"]   * combined_wide$ADV +
  coefs["AUX"]   * combined_wide$AUX +
  coefs["CCONJ"] * combined_wide$CCONJ +
  coefs["DET"]   * combined_wide$DET +
  coefs["NOUN"]  * combined_wide$NOUN +
  coefs["NUM"]   * combined_wide$NUM +
  coefs["PART"]  * combined_wide$PART +
  coefs["PRON"]  * combined_wide$PRON +
  coefs["PROPN"] * combined_wide$PROPN +
  coefs["PUNCT"] * combined_wide$PUNCT +
  coefs["SCONJ"] * combined_wide$SCONJ +
  coefs["SYM"]   * combined_wide$SYM +
  coefs["VERB"]  * combined_wide$VERB

combined_wide$prob <- 1 / (1 + exp(-combined_wide$logit))

# Klassenvorhersage mit aktuellem Cutoff
combined_wide$pred_success <- ifelse(combined_wide$prob <= prob_cutoff, 0, 1)

tab <- table(combined_wide$pred_success, combined_wide$Category)

# Sicheres Auslesen aus der Konfusionsmatrix
TN <- ifelse("Failed" %in% colnames(tab) && "0" %in% rownames(tab), tab["0", "Failed"], 0)
FN <- ifelse("Success" %in% colnames(tab) && "0" %in% rownames(tab), tab["0", "Success"], 0)
TP <- ifelse("Success" %in% colnames(tab) && "1" %in% rownames(tab), tab["1", "Success"], 0)
FP <- ifelse("Failed" %in% colnames(tab) && "1" %in% rownames(tab), tab["1", "Failed"], 0)

# Metriken berechnen
SENS <- round(ifelse((TP + FN) == 0, NA, TP / (TP + FN)), 4)
SPEC <- round(ifelse((TN + FP) == 0, NA, TN / (TN + FP)), 4)
ACCU <- round(ifelse((TP + TN + FP + FN) == 0, NA, (TP + TN) / (TP + TN + FP + FN)), 4)
PPV <- round(ifelse((TP + FP) == 0, NA, TP / (TP + FP)), 4)
NPV <- round(ifelse((TN + FN) == 0, NA, TN / (TN + FN)), 4)
F1  <- round(ifelse((PPV + SENS) == 0, NA, 2 * (PPV * SENS) / (PPV + SENS)), 4)
J   <- round(ifelse(is.na(SENS) || is.na(SPEC), NA, SENS + SPEC - 1), 4)

YEAR <- years$year[i]

metrics_row <- data.frame(
  CUTOFF = prob_cutoff,
  YEAR = YEAR,
  TN = TN,
```

```
      FN = FN,
      TP = TP,
      FP = FP,
      SENS = SENS,
      SPEC = SPEC,
      ACCU = ACCU,
      PPV = PPV,
      NPV = NPV,
      F1 = F1,
      J = J
    )
    quality <- rbind(quality, metrics_row)
    threshold_results <- rbind(threshold_results, metrics_row)
  }
  all_quality <- rbind(all_quality, quality)
}
```



Beurteilung des Sprachmodells

Für die Jahre 2017 bis 2023 wurden hier die Prognosen des Erfolges auf Grundlage des Sprachmodells von 2016 bestimmt. Für den Vergleich der Vorhersage und dem tatsächlichen Ergebnis existieren verschiedene Relationen, die in folgender Tabelle gegenüber gestellt werden:

	tatsächlich erfolgreich	tatsächlich erfolglos
vorausgesagt als erfolgreich	(a) True Positive	(b) False Positive
vorausgesagt als nicht erfolgreich	(c) False Negativ	(d) True Negativ

Sensitivität (Sensitivity)= a/(a+c)

Die Sensitivität (auch als True Positive Rate oder Recall bezeichnet; Baratloo et al. 2015, p.48) gibt an, wie gut ein Test, ein Modell oder ein Algorythmus ein konkretes positives Ergebnis richtig erkennt, also erfolgreiche Projekte als erfolgreich, Kranke als krank oder Spam als Spam.

$$Sensitivity = \frac{TP}{TP + FN}$$

Sie gibt also an, wie gut die tatsächlich Positiven von einem Test erkannt werden. Bei einer Sensitivität $>$ 0.90 werden nahezu alle Positiven erkannt.

Sensitivität	Interpretation
> 0.9	sehr gut
0.8 - 0.9	gut
0.6 - 0.8	mittelmäßig
< 0.6	schwach

Spezifität (Specificity)

Die Spezifität (auch als True Negative Rate bezeichnet; Baratloo et al. 2015, p.48) gibt an, wie gut ein Test, ein Modell oder ein Algorythmus ein konkretes negatives Ergebnis richtig erkennt, also erfolglose Projekte als nicht erfolgreich, Gesunde als gesund oder Emails als relevant. Sie gibt also an, wie gut die tatsächlich Negativen von einem Test erkannt werden. Eine Spezifizität > 0.90 löst nahezu keinen Fehlalarm aus.

$$Spezifität = \frac{TN}{TN + FP}$$

Spezifität	Interpretation
> 0.9	sehr gut
0.8 - 0.9	gut
0.6 - 0.8	mittelmäßig
< 0.6	schwach

Die Interpretation von $1 - Spezifität$ weist auf die False Positive Rate (FPR) hin, also der Anteil der fälschlicherweise als positiv klassifizierten, aber tatsächlich negativen Fälle.

Receiver Operating Characteristic (ROC) Kurve

Die ROC-Kurve misst die Fähigkeit eines Modells, zwischen zwei Klassen zu unterscheiden, indem sie True Positive Rate (TPR) gegen False Positive Rate (FPR) bei verschiedenen Thresholds vergleicht.

Die ROC-Kurve zeigt, wie gut ein Klassifikationsmodell zwischen zwei Klassen unterscheidet. Sie basiert auf der Analyse von True Positive Rate (TPR) und False Positive Rate (FPR) bei verschiedenen Schwellenwerten. True Positive Rate (TPR) = Sensitivität = Anteil der korrekt als positiv erkannten Fälle False Positive Rate (FPR) = 1 - Spezifität = Anteil der fälschlich als positiv erkannten negativen Fälle

Die ROC-Kurve stellt TPR gegen FPR bei verschiedenen Thresholds (Entscheidungsgrenzen) grafisch dar:

X-Achse: FPR (False Positive Rate) Y-Achse: TPR (True Positive Rate) Das Modell liefert oft Wahrscheinlichkeiten (z. B. 0 bis 1), und durch Anpassen des Thresholds kann man entscheiden, ab wann ein Fall als „positiv“ gilt.

Eine perfekte Klassifikation hätte eine ROC-Kurve, die sofort nach oben und dann nach rechts verläuft (Fläche unter der Kurve = 1). Eine zufällige Klassifikation verläuft als Diagonale von (0,0) nach (1,1) (Fläche = 0,5). Je näher die Kurve an der oberen linken Ecke, desto besser das Modell.

Die Fläche unter der ROC-Kurve, die AUC (Area Under Curve), gibt einen summarischen Leistungswert:

AUC = 1: perfekte Klassifikation AUC = 0.5: reines Raten AUC $<$ 0.5: schlechter als Zufall (Modell vertauscht Klassen)

Accuracy

Accuracy (dt. Genauigkeit) ist eine der bekanntesten, aber auch am häufigsten missverstandenen Metriken in der Klassifikation. “The accuracy of a test is its ability to differentiate the patient and healthy cases correctly. To estimate the accuracy of a test, we should calculate the proportion of true positive and true negative in all evaluated cases. Mathematically, this can be stated as” (Baratloo et al. 2015, p.48):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Die Genauigkeit eignet sich nur dann, wenn die Klassen ausgewogen sind, als fairer Gesamteinschätzung der Modellgüte.

Genauigkeit	Interpretation
90 - 100%	sehr gut (mglw. ein Hinweis auf Overfitting
80 - 90% - 0.9	gut (bei gleichmäßig verteilten Klassen)
70 - 80 %	solide (bei gleichmäßig verteilten Klassen)
rund 50%	nicht besser als Zufall

Positiver Prädiktiver Wert (PPV; precision) = a/(a+b)

Das führt auch zu der Frage, wie viele der als positiv klassifizierten Fälle wirklich positiv sind (Safari et al. 2015; p.87). Im medizinischen Bereich ist dafür die Bezeichnung positiver prädiktiver Wert (PPV) geläufig und “bezeichnet den Anteil der Kranken unter den Verdachtsfällen, also den Anteil der richtig positiven von allen positiven Testergebnissen” (Klemperer, Sozialmedizin - Public Health; S.189). Im Machine Learning wird der Wert gewöhnlich als Precision bezeichnet

$$PPV = \frac{TP}{TP + FP}$$

Präzision (PPV)	Interpretation
> 0.9	sehr zuverlässig
0.7 - 0.9	ziemlich gut
0.5 - 0.7	akzeptabel, aber zu viele Fehlalarme
< 0.5	mehr Fehlalarm als Treffer

Negativer Prädiktiver Wert (NPV) = d/(d+c)

Selterner und im machine learning wenig bekannt, ist der negative prädiktive Wert NPV. Damit wird angegeben, wie viele der als negativ klassifizierten Fälle wirklich negativ sind (Safari et al. 2015; p.87), also z.B. der Anteil der Gesunden unter den Ausschlussfällen (negativen Testergebnissen)

$$NPV = \frac{TN}{TN + FN}$$

Negativer prädiktiver Wert (NPV)	Interpretation
> 0.9	sehr zuverlässig
0.7 - 0.9	ziemlich gut
0.5 - 0.7	akzeptabel, aber zu viele falsche Entwarnung
< 0.5	mehr falsche als richtige Entwarnung

Kombinierte Indizes

Es gibt keine "perfekten" Kriterien zur Bewertung – sie hängen stark vom Anwendungsfall ab: wenn falsche Alarme teuer sind (z. B. automatisierte Diagnostik, Spam-Filter) oder bei Klassifikationen mit geringer Prävalenz, (d.h. viele Positive könnten falsch sein), hat die Präzision eine enorme Bedeutung.

Anwendung	Orientierung
Krebs-Screening	Hohe Sensitivität
HIV-Bestätigungstest	Hohe Spezifität
Spam-Filter	Gute Präzision (wenig falsch positive)
Kreditrisiko	Balanciert: alle drei relevant

F1_score

Der F1-Score ist eine Metrik zur Bewertung von Klassifikationsmodellen, die das Gleichgewicht zwischen Präzision und Sensitivität misst. Er wird häufig im maschinellen Lernen angewandt und ist besonders bei seltenen Ereignissen (z.B.Betrugserkennung) sinnvoll. Die Klassenverteilung ist dann unausgewogen (z.B. 95% negativ, 5% positiv). Wenn ein starkes Ungleichgewicht zwischen Präzision und Sensitivität auftritt, "bestraft" der F1-Score eine hohe Präzision bei gleichzeitig geringer Sensitivität.

$$F1Score = 2 \cdot \frac{Präzision \cdot Sensitivität}{Präzision + Sensitivität}$$

F1-Score	Interpretation
> 0.9	exzellent
0.7 - 0.9	gut
0.5 - 0.7	mittelmäßig
< 0.5	schwach

Youden-Index (J) – Einfaches Maß zur Cutoff-Optimierung

Eine weitere wichtige Kennzahl zur Bewertung von Klassifikationsmodellen – besonders im medizinischen Bereich oder allgemein bei binärer Klassifikation ist der Youden-Index (J). Als Index vereint J Sensitivität und Spezifität und ist nützlich, um den optimalen Schwellenwert (Threshold) für binäre Entscheidungen (probabilistischen Klassifikationen) zu finden.

Ein hoher Wert von J bedeutet, dass wenig tatsächlich Positive übersehen werden (hohe Sensitivität) und wenige tatsächlich Negative fälschlich als positiv gelten (hohe Spezifität).

Wenn die Modellparameter danach bestimmt werden, dass J seinen maximal Wert annimmt, ist daher der beste Punkt, um das Modell zu betreiben, wenn beide Fehlerarten gleich gewichten werden sollen.

$$Youden(J) = Sensitivität + Spezifität - 1$$

Youden (J)	Interpretation
1	perfekt
0	nicht besser als Zufall
<0	schlechter als Zufall

Ergebnisse

Im Datenframe "quality" sind jetzt für die einzelnen Jahre (in Zeilen) die Kriterien der Vorhersage gespeichert. Der zugrundeliegende cutoff-Wert war p=0.50.

```
knitr::kable(
  #head(quality[, 1:12], ), booktabs = TRUE,
  quality[, 2:13]
  )
```

YEAR	TN	FN	TP	FP	SENS	SPEC	ACCU	PPV	NPV	F1	J
2017	11186	11196	1477	625	0.1165	0.9471	0.5172	0.7027	0.4998	0.1999	0.0636
2018	8573	11367	1508	506	0.1171	0.9443	0.4592	0.7488	0.4299	0.2025	0.0614
2019	7779	16329	2210	433	0.1192	0.9473	0.3734	0.8362	0.3227	0.2087	0.0665
2020	6497	16553	2206	308	0.1176	0.9547	0.3404	0.8775	0.2819	0.2074	0.0723
2021	4795	15941	2267	229	0.1245	0.9544	0.3040	0.9083	0.2312	0.2190	0.0789
2022	1956	8123	1118	91	0.1210	0.9555	0.2723	0.9247	0.1941	0.2140	0.0765
2023	117	512	41	6	0.0741	0.9512	0.2337	0.8723	0.1860	0.1366	0.0253

```
colMeans(quality[, 7:13])
```

```
##      SENS      SPEC      ACCU       PPV       NPV        F1         J
## 0.1128571 0.9506429 0.3571714 0.8386429 0.3065143 0.1983000 0.0635000
```

TN=True Negative, FN=False Negative, TP=True Positive, FP=False Positive, SENS=Sensitivity, SPEC=Specifity, ACCU=Accuracy, PPV=Positive Predicted Value (Precision), NPV=Negative Predicted Value F1=F1-Score, J=Youle-Index

Suche nach dem J_{max} bzw. dem optimalen Schwellenwert p_{best}

```
library(dplyr)

# Annahme: threshold_results enthält die grobe Übersicht (aus vorherigem Schritt)

# Grobe Optimierung: bester J pro Jahr
best_cutoffs_grob <- threshold_results %>%
  group_by(YEAR) %>%
  filter(J == max(J, na.rm = TRUE)) %>%
  select(YEAR, grob_cutoff = CUTOFF, J_grob = J)

# Vorbereitung: Ergebnis-Datenframe
fine_results <- data.frame()

# Schleife: Für jedes Jahr feinen Bereich um besten groben Cutoff prüfen
for (i in 1:nrow(best_cutoffs_grob)) {
  jahr <- best_cutoffs_grob$YEAR[i]
  grob_cutoff <- best_cutoffs_grob$grob_cutoff[i]

  # Neues Intervall um den groben Cutoff, z. B. ±0.05 in 0.01er-Schritten
```

```r
fine_cutoffs <- seq(max(0, grob_cutoff - 0.05), min(1, grob_cutoff + 0.05), by = 0.01)

# Daten laden & vorbereiten
success_data <- read.csv(paste0(folder, jahr, "_success_all_upos_counts.csv"))
failed_data  <- read.csv(paste0(folder, jahr, "_fail_all_upos_counts.csv"))
success_data$Category <- "Success"
failed_data$Category <- "Failed"

combined_data <- bind_rows(success_data, failed_data)
combined_data$Freq <- combined_data$Freq * 100 / combined_data$Tokens

combined_wide <- combined_data %>%
  pivot_wider(names_from = Var2, values_from = Freq)
combined_wide[, 5:21][is.na(combined_wide[, 5:21])] <- 0
combined_wide$success <- ifelse(combined_wide$Category == "Success", 1, 0)

# Logit & Wahrscheinlichkeiten berechnen
combined_wide$logit <- coefs["Intercept"] +
  coefs["ADJ"]*combined_wide$ADJ +
  coefs["ADV"]*combined_wide$ADV +
  coefs["AUX"]*combined_wide$AUX +
  coefs["CCONJ"]*combined_wide$CCONJ +
  coefs["DET"]*combined_wide$DET +
  coefs["NOUN"]*combined_wide$NOUN +
  coefs["NUM"]*combined_wide$NUM +
  coefs["PART"]*combined_wide$PART +
  coefs["PRON"]*combined_wide$PRON +
  coefs["PROPN"]*combined_wide$PROPN +
  coefs["PUNCT"]*combined_wide$PUNCT +
  coefs["SCONJ"]*combined_wide$SCONJ +
  coefs["SYM"]*combined_wide$SYM +
  coefs["VERB"]*combined_wide$VERB

combined_wide$prob <- 1 / (1 + exp(-combined_wide$logit))

# Loop über feine Cutoffs
for (cutoff in fine_cutoffs) {
  combined_wide$pred_success <- ifelse(combined_wide$prob <= cutoff, 0, 1)
  tab <- table(combined_wide$pred_success, combined_wide$Category)

  # Fehlerresistente Konfusionsmatrix
  TN <- ifelse("Failed" %in% colnames(tab) && "0" %in% rownames(tab), tab["0", "Failed"], 0)
  FN <- ifelse("Success" %in% colnames(tab) && "0" %in% rownames(tab), tab["0", "Success"], 0)
  TP <- ifelse("Success" %in% colnames(tab) && "1" %in% rownames(tab), tab["1", "Success"], 0)
  FP <- ifelse("Failed" %in% colnames(tab) && "1" %in% rownames(tab), tab["1", "Failed"], 0)

  SENS <- round(ifelse((TP + FN) == 0, NA, TP / (TP + FN)), 4)
  SPEC <- round(ifelse((TN + FP) == 0, NA, TN / (TN + FP)), 4)
  J <- round(ifelse(is.na(SENS) || is.na(SPEC), NA, SENS + SPEC - 1), 4)

  fine_results <- rbind(fine_results, data.frame(
    YEAR = jahr,
    CUTOFF = cutoff,
```

```
      SENS = SENS,
      SPEC = SPEC,
      J = J
    ))
  }
}

# Bestes Ergebnis pro Jahr finden
best_cutoffs_fine <- fine_results %>%
  group_by(YEAR) %>%
  filter(J == max(J, na.rm = TRUE)) %>%
  select(YEAR, optimal_cutoff = CUTOFF, optimal_J = J)

print(best_cutoffs_fine)
```

```
## # A tibble: 7 x 3
## # Groups:   YEAR [7]
##   YEAR  optimal_cutoff optimal_J
##   <chr>          <dbl>     <dbl>
## 1 2023            0.24     0.378
## 2 2017            0.26     0.289
## 3 2018            0.25     0.281
## 4 2019            0.25     0.301
## 5 2020            0.26     0.292
## 6 2021            0.27     0.283
## 7 2022            0.27     0.271
```

Der optimale Schwellenwert ist laut Youden-Index (J): p=0.257 ist laut Youde-Index besser!!!

Verwendung des optimalen Schwellenwertes

Unter Berücksichtigung des J-optimierten Schwellenwertes für die Klassifizierung ergeben sich die folgenden Metriken.

```
# Pakete laden
library(dplyr)
library(tidyr)
library(ggplot2)
library(readr)


# übernimmt den optimalen Schwellenwert bei maximalen J als Mittelwert der Jahre
prob_cutoff =mean(best_cutoffs_fine$optimal_cutoff)

years <- data.frame(year = c("2017", "2018", "2019", "2020", "2021", "2022", "2023" ))
folder <- "~/Markdown/Kickstarter_Ling/data/kickstarter/"

# ein datenframe erstellen, um darin die Gütekriterien zu speichern
quality <- data.frame(
  YEAR = character(),
  TN = numeric(),
  FN = numeric(),
```

```
  TP = numeric(),
  FP = numeric(),
  SENS = numeric(),
  SPEC = numeric(),
  ACCU = numeric(),
  PPV = numeric(),
  NPV = numeric(),
  F1 = numeric(),
  J= numeric()
)

library(pROC)
library(tidyverse)  # für bind_rows, pivot_wider etc.

# Ergebnis-Tabellen vorbereiten
all_quality <- data.frame()
threshold_results <- data.frame()


par(mfrow = c(nrow(years)/2,2))  # eine Zeile, je Jahr ein Plot

# ----------------------------------------
# Metriken bei dem besten Cutoffs berechnen
# ----------------------------------------
  quality <- data.frame()

  for (i in 1:nrow(years)) {
    # Daten wie zuvor
    success_data <- read.csv(paste0(folder, years$year[i], "_success_all_upos_counts.csv"))
    failed_data  <- read.csv(paste0(folder, years$year[i], "_fail_all_upos_counts.csv"))

    success_data$Category <- "Success"
    failed_data$Category  <- "Failed"

    combined_data <- bind_rows(success_data, failed_data)
    combined_data$Freq <- combined_data$Freq * 100 / combined_data$Tokens

    combined_wide <- combined_data %>%
      pivot_wider(names_from = Var2, values_from = Freq)

    combined_wide[, 5:21][is.na(combined_wide[, 5:21])] <- 0
    combined_wide$success <- ifelse(combined_wide$Category == "Success", 1, 0)

    combined_wide$logit <- coefs["Intercept"] +
      coefs["ADJ"]   * combined_wide$ADJ +
      coefs["ADV"]   * combined_wide$ADV +
      coefs["AUX"]   * combined_wide$AUX +
      coefs["CCONJ"] * combined_wide$CCONJ +
      coefs["DET"]   * combined_wide$DET +
      coefs["NOUN"]  * combined_wide$NOUN +
      coefs["NUM"]   * combined_wide$NUM +
      coefs["PART"]  * combined_wide$PART +
      coefs["PRON"]  * combined_wide$PRON +
```

```
    coefs["PROPN"] * combined_wide$PROPN +
    coefs["PUNCT"] * combined_wide$PUNCT +
    coefs["SCONJ"] * combined_wide$SCONJ +
    coefs["SYM"]   * combined_wide$SYM +
    coefs["VERB"]  * combined_wide$VERB

  combined_wide$prob <- 1 / (1 + exp(-combined_wide$logit))

  # Klassenvorhersage mit aktuellem Cutoff
  combined_wide$pred_success <- ifelse(combined_wide$prob <= prob_cutoff, 0, 1)

  tab <- table(combined_wide$pred_success, combined_wide$Category)

  # Sicheres Auslesen aus der Konfusionsmatrix
  TN <- ifelse("Failed" %in% colnames(tab) && "0" %in% rownames(tab), tab["0", "Failed"], 0)
  FN <- ifelse("Success" %in% colnames(tab) && "0" %in% rownames(tab), tab["0", "Success"], 0)
  TP <- ifelse("Success" %in% colnames(tab) && "1" %in% rownames(tab), tab["1", "Success"], 0)
  FP <- ifelse("Failed" %in% colnames(tab) && "1" %in% rownames(tab), tab["1", "Failed"], 0)

  # Metriken berechnen
  SENS <- round(ifelse((TP + FN) == 0, NA, TP / (TP + FN)), 4)
  SPEC <- round(ifelse((TN + FP) == 0, NA, TN / (TN + FP)), 4)
  ACCU <- round(ifelse((TP + TN + FP + FN) == 0, NA, (TP + TN) / (TP + TN + FP + FN)), 4)
  PPV <- round(ifelse((TP + FP) == 0, NA, TP / (TP + FP)), 4)
  NPV <- round(ifelse((TN + FN) == 0, NA, TN / (TN + FN)), 4)
  F1  <- round(ifelse((PPV + SENS) == 0, NA, 2 * (PPV * SENS) / (PPV + SENS)), 4)
  J   <- round(ifelse(is.na(SENS) || is.na(SPEC), NA, SENS + SPEC - 1), 4)

  YEAR <- years$year[i]

  metrics_row <- data.frame(
    CUTOFF = prob_cutoff,
    YEAR = YEAR,
    TN = TN,
    FN = FN,
    TP = TP,
    FP = FP,
    SENS = SENS,
    SPEC = SPEC,
    ACCU = ACCU,
    PPV = PPV,
    NPV = NPV,
    F1 = F1,
    J = J
  )

  quality <- rbind(quality, metrics_row)
  threshold_results <- rbind(threshold_results, metrics_row)
}

all_quality <- rbind(all_quality, quality)
```

```
knitr::kable(
  #head(quality[, 1:12], ), booktabs = TRUE,
  all_quality[, 2:13]
  )
```

YEAR	TN	FN	TP	FP	SENS	SPEC	ACCU	PPV	NPV	F1	J
2017	7071	3954	8719	4740	0.6880	0.5987	0.6449	0.6478	0.6414	0.6673	0.2867
2018	5256	3872	9003	3823	0.6993	0.5789	0.6495	0.7019	0.5758	0.7006	0.2782
2019	4861	5421	13118	3351	0.7076	0.5919	0.6721	0.7965	0.4728	0.7494	0.2995
2020	4024	5619	13140	2781	0.7005	0.5913	0.6714	0.8253	0.4173	0.7578	0.2918
2021	2887	5315	12893	2137	0.7081	0.5746	0.6792	0.8578	0.3520	0.7758	0.2827
2022	1164	2791	6450	883	0.6980	0.5686	0.6745	0.8796	0.2943	0.7783	0.2666
2023	86	182	371	37	0.6709	0.6992	0.6760	0.9093	0.3209	0.7721	0.3701

TN=True Negative, FN=False Negative, TP=True Positive, FP=False Positive, SENS=Sensitivity, SPEC=Specifity, ACCU=Accuracy, PPV=Positive Predicted Value (Precision), NPV=Negative Predicted Value F1=F1-Score, J=Youle-Index

Unter Verwendung der optimalen Schwelle verbessern sich die Sensitivität von 0.11 (schwach) auf 0.70 (mittelmäßig), die Spezifität fällt von 0.95 (sehr gut) auf 0.60 (mittelmäßig), die Genauigkeit verbessert sich von 0.36 (nicht besser als Zufall) auf 0.67 (nahezu solide), die PPV bleibt ziemlich gut (0.80), die NPV bleibt gering (0.44; mehr falsche als richtige Entwartungen) und der F1_Score steigt auf 0.30 (besser als Zufall).

Dieses Ergebnis kann als ernstzunehmende Ergänzung zur rein inhaltlichen Auszählung von konstruktspezifischen Wortlisten angesehen werden.