Bachelorarbeit - Prompt Engineering Handbook

Imports

Importing “AER” library to use “tobit regression”-models:

```
#install.packages("AER") uncomment if the package isnt already installed
library(AER)
```

```
Lade nötiges Paket: car
```

```
Lade nötiges Paket: carData
```

```
Lade nötiges Paket: lmtest
```

```
Lade nötiges Paket: zoo
```

```

Attache Paket: 'zoo'
```

```
Die folgenden Objekte sind maskiert von 'package:base':
```

```
    as.Date, as.Date.numeric
```

```
Lade nötiges Paket: sandwich
```

```
Lade nötiges Paket: survival
```

Importing “readxl” library so that Excel-sheets can be read:

```r
#install.packages("readxl") uncomment if the package isnt already installed
library(readxl)
```

Datamanagement

The basis for the analysis done in this thesis is an experimental phase using GitHub Copilot to explore limitations and possibilities for artificial creation of unit-tests. All the data from the experiments were saved into an excel sheet (1 sheet per model). The excel sheets are now used to create the prompt engineering handbook where the different prompting strategies are analysed and concrete actions are recommended based on the findings.

Here the excel sheets are read in. There is average sheets containing the average values for a quick overview as well as more detailed sheets containing all results for all prompting strategies and models that will be used for creating the comprehensive analysis:

```r
# List of sheets - averageValues
averageValues <- c("avgOpenAI", "avgClaude", "avgGemini")

# Looping through each sheet and assigning it to a variable with the same name
for (sheet in averageValues) {
  df <- read_excel("DatenPrompts.xlsx", sheet = sheet)
  df$DataType <- "AverageData"
  assign(sheet, df)
}

# List of sheets - actual data
data <- c("OpenAI", "Gemini", "Claude")

# Looping through each sheet and assigning it to a variable with the same name
for (sheet in data) {
  df <- read_excel("DatenPrompts.xlsx", sheet = sheet)
  df$DataType <- "ActualData"
  assign(sheet, df)
}
```

Then we can combine the different sheets into one single dataframe:

```r
# Combine your list of sheet names
allSheets <- c(averageValues, data)

# Extract each data frame into a list
```

```
sheets_list <- lapply(allSheets, get)
```

```
allSheets <- c(averageValues, data)
allData <- do.call(rbind, lapply(allSheets, get))
```

In the next step relevant columns will be transformed into Factors to make them available for statistical analysis like t-tests or regression and afterwards all sheets will be transformed into one, single dataframe to make the analysis easier:

```
allData$Model   <- as.factor(allData[[9]])
allData$Detail  <- factor(allData[[3]], levels = c("low", "medium", "high"))
allData$Length  <- factor(allData[[4]], levels = c("low", "medium", "high"))
allData$Context <- factor(allData[[5]], levels = c("low", "medium", "high"))
```

Additionally, the factors Model, Detail and Context will be combined into a single, composite variable to make the prompting strategies available for statistical analysis.

```
allData$Strategy <- with(allData, interaction(Detail, Context, Length, sep = "_"))

#use droplevels to ensure we only have the relevant strategies (high high high, medium med
allData$Strategy <- droplevels(allData$Strategy)
```

Now we can create a subgroup and use either the average values or the actual data for the analysis or plotting:

```
avgData <- subset(allData, DataType == "AverageData")

mainData <- subset(allData, DataType == "ActualData")
```

After reading and properly setting up all data, we can start with our analysis.

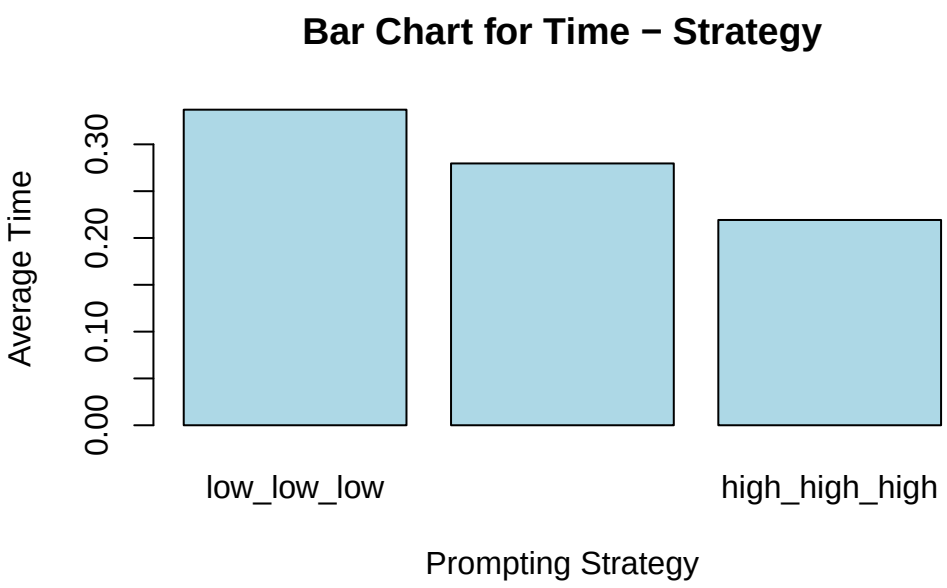
Analysis

Visualization

At first there will be a short visualization to gain a first visual impression based on our average values

```
aggDataTime <- aggregate(Time ~ Strategy, data = avgData, FUN = mean)
```

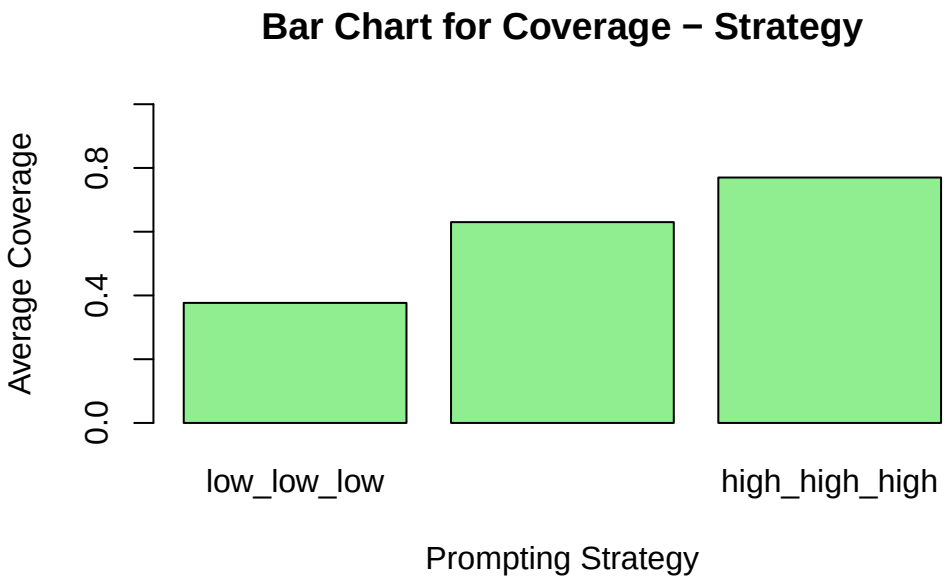
```r
#barplot
barplot(aggDataTime$Time,
        names.arg = aggDataTime$Strategy,
        col = "lightblue",
        border = "black",
        main = "Bar Chart for Time - Strategy",
        xlab = "Prompting Strategy",
        ylab = "Average Time")
```



The barplot shows a clear trend regarding prompting strategies: The better the prompting strategy, the lower the cycle time.

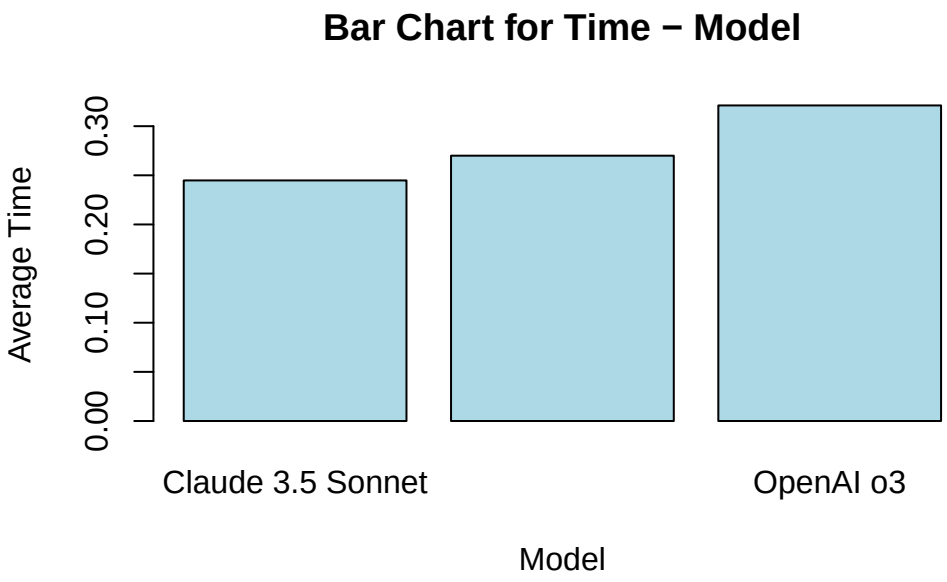
```r
aggDataCoverage <- aggregate(`Coverage%` ~ Strategy, data = avgData, FUN = mean)

#barplot
barplot(aggDataCoverage$`Coverage%`,
        names.arg = aggDataCoverage$Strategy,
        col = "lightgreen",
        border = "black",
        main = "Bar Chart for Coverage - Strategy",
        xlab = "Prompting Strategy",
```

```
        ylab = "Average Coverage",
        ylim = c(0, 1))
```



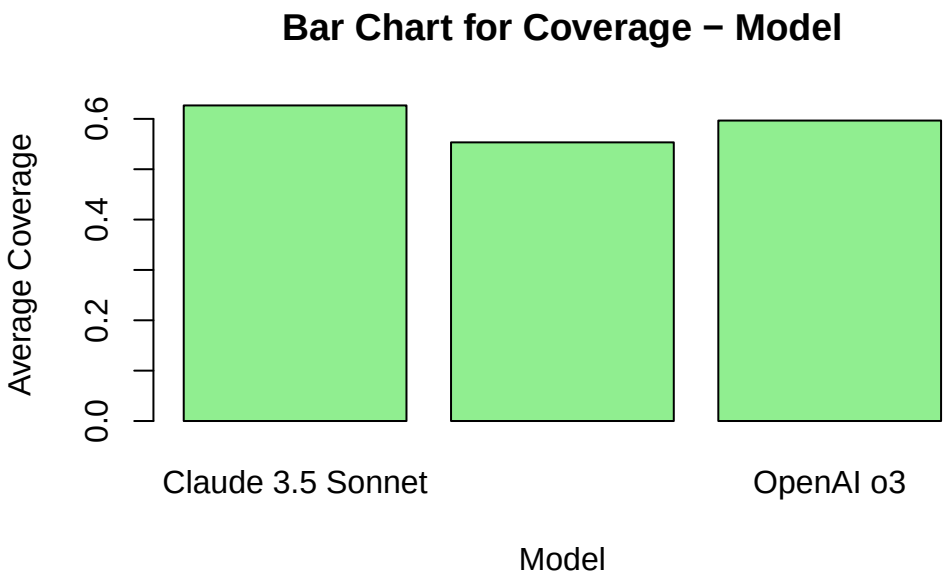
The bar plot shows a clear trend regarding prompting strategies: The better the prompting strategy, the higher the test coverage. The average test coverage doubles going from low to high.

```
aggDataTimeModel <- aggregate(Time ~ Model, data = avgData, FUN = mean)

#barplot
barplot(aggDataTimeModel$Time,
        names.arg = aggDataTimeModel$Model,
        col = "lightblue",
        border = "black",
        main = "Bar Chart for Time - Model",
        xlab = "Model",
        ylab = "Average Time")
```



The barplot show the average time per model. OpenAI has on average the highest time, followed by Claude 3.5 Sonnet. Google Gemini 2.0 Flash has the lowest average time.

```r
aggDataCoverageModel <- aggregate(`Coverage%` ~ Model, data = avgData, FUN = mean)

#barplot
barplot(aggDataCoverageModel$`Coverage%`,
        names.arg = aggDataCoverageModel$Model,
        col = "lightgreen",
        border = "black",
        main = "Bar Chart for Coverage - Model",
        xlab = "Model",
        ylab = "Average Coverage")
```



The barplot show the average coverage per model. Claude 3.5 Sonnet has the highest test coverage, closely followed by OpenAI o3. Google Gemini 2.0 Flash has the lowest test coverage.

Discussion of average values

The bar plots clearly showed that while analyzing each outcome variable independently, the combination must also be considered. Because regarding time Google Gemini is preferable, regarding test coverage it is actually the worst model.

Discussion of statistical methods used:

The following chapters contains a small discussion regarding the implemented statistical analysis methods and why they were used.

Linear Regression

Linear regression models allow to analyse how the outcome variables (in the context of this thesis "test coverage", „cycle time" and "passed") are influenced by prompting strategies (composed of the three independent variables: "Prompt detail", "Quality of the prompt", and "Level of context") as well as different AI models. (Kutner et al., 2005). In the context of this thesis, using linear regression is highly informative because it gives insight how each prompting strategy as well as models contribute to the out-come variables. This enables future users to opt for better strategies.

Anova

Ideal for comparing means across multiple groups (e.g., different AI models or dif-ferent prompting strategies) for a single outcome variable. This can lead to a better understanding if variability regarding the outcome is statistically significant and can be attributed to a specific predictor (Kutner et al., 2005). In the context of this thesis, Anova helps identifying which prompting strategies and/or models lead to better results regarding the outcome variables.

Manova (Multivariate Anova)

Manova allows to analyse the effects of each predictor for the outcome variables at the same time. This is particularly useful when trying to analyse how independent variables interact with more than outcome variable. Through this, interactions could be captured that might have been overlooked when looking at each outcome individ-ually (Huberty and Olejnik, 2006).

This is highly useful as it shows how different prompting strategies affect **both** cycle time and test coverage.

T-Tests

T-tests are useful for comparisons between groups based on the mean. This can be used to examine the influence of single predictors on the outcome variables and dis-cuss the differences between distinct levels of each predictor (Kim, 2015).

This can be particularly useful for this thesis as it shows whether changing specific aspects of the prompting strategy (level of context) can influence the outcomes.

Correlation Analysis

Correlation analysis explores the relationships among independent variable and how strongly each predictor is associated with the outcome variables (Janse et al., 2021). In the context of this thesis, this is valuable as it highlights how prompting strategies are related to high coverage and/or low cycle time.

All in, these methods are used in an attempt to provide a comprehensive overview and statistical evaluation of prompting strategies and their importance for the artificial generation of unit-tests. This is important for developing a profound prompt engineering handbook. Also in the context of this thesis not a complete handbook can be created but based on the findings a relatively extensive prototype should be created and serve as a basis for continous research.

Performing (Statistical) Analysis

Time Analysis

The first part will cover the outcome variable “cycle time” and will perform the mentioned statistical tests to gain insight into the possible relationship between cycle time and prompting

strategy and/or model. At first, there will be a correlation analysis for a broad overview. Then followed by anova to gain a more detailed look into the effects across groups. Subsequently, there will be T-Tests to examine each groups more closely. Last but not least, there will be the linear regression analysis to gain more insight into how each strategy influences the generation of unit-tests.

Correlation - Time

In the first step, we will have a look at Correlation as a first exploratory phase. This should provide insights into a general direction and if it is even worth to explore further.

Prompting Strategy

```r
mainData$StrategyNumeric <- as.numeric(mainData$Strategy)

cor_test_result <- cor.test(mainData$Time, mainData$StrategyNumeric, method = "pearson")
print(cor_test_result)
```

```
    Pearson's product-moment correlation

data:  mainData$Time and mainData$StrategyNumeric
t = -3.5459, df = 133, p-value = 0.0005408
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.4409525 -0.1314614
sample estimates:
       cor
-0.2938909
```

The correlation analysis confirms that there is a strong relationship between cycle time and prompting strategy. A correlation coefficient of -0.29 suggests a slight moderate negative relationship (Akoglu, 2018). The p-value of 0.0005408 as well as the 95% confidence interval confirm the statistical significance of this relationship.

Although this may seem low, it already works as a good indicator for the overall hypothesis that good prompting strategies lower the cycle time (or time needed to create unit-tests which then lowers cycle time).

ANOVA - Time

Now, we will use anova to gain a more detailed look into the effects across groups.

Prompting Strategy

```
aov_modelTime <- aov(Time ~ Strategy, data = mainData)
summary(aov_modelTime)
```

```
             Df Sum Sq Mean Sq F value  Pr(>F)
Strategy      2  0.347 0.17356   6.254 0.00254 **
Residuals   132  3.663 0.02775
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The result of the anova test (more specifically the small p-value of 0.00254) shows clearly that there are statistical significant differences between different groups of strategy. Hence, the anova test strongly suggests that prompting strategies is an important variable for predicting time and supports the idea that good prompting techniques are necessary to leverage GitHub Copilot more efficiently and reduce cycle time.

TukeyHSD

The following code performs a post-hoc analysis on the anova model using TukeyHSD in order to see the importance of each level for the predictor strategy. Tukey is a widely used follow-up test for anova models (Abdi and Williams, 2010).

```
TukeyHSD(aov_modelTime, "Strategy")
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Time ~ Strategy, data = mainData)

$Strategy
                                          diff        lwr         upr
medium_medium_medium-low_low_low    -0.05703704 -0.1402859  0.02621182
high_high_high-low_low_low          -0.12407407 -0.2073229 -0.04082521
high_high_high-medium_medium_medium -0.06703704 -0.1502859  0.01621182
                                        p adj
medium_medium_medium-low_low_low    0.2391154
high_high_high-low_low_low          0.0016337
high_high_high-medium_medium_medium 0.1401898
```

According to the TukeyHSD test, only the comparison between high_high_high and low_low_low is statistically significant. However, for this it needs to be taken into consideration that time is not the only factor to be considered.

Model

```
aov_model <- aov(Time ~ Model, data = mainData)
summary(aov_model)
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
Model         2  0.101 0.05040   1.702  0.186
Residuals   132  3.909 0.02962
```

The result of the anova test (more specifically the p-value of 0.186) shows clearly that there are no statistical significant differences between different models. The anova test therefore does not support the idea that different models lead to an improved cycle time.

TukeyHSD

The following code performs a post-hoc analysis on the anova model using TukeyHSD in order to see the importance of each level for the predictor model.

```
TukeyHSD(aov_model, "Model")
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Time ~ Model, data = mainData)

$Model
                                                diff         lwr       upr
Google Gemini 2.0 Flash-Claude 3.5 Sonnet 0.01407407 -0.07192835 0.1000765
OpenAI o3-Claude 3.5 Sonnet               0.06370370 -0.02229872 0.1497061
OpenAI o3-Google Gemini 2.0 Flash         0.04962963 -0.03637279 0.1356321
                                              p adj
Google Gemini 2.0 Flash-Claude 3.5 Sonnet 0.9204569
OpenAI o3-Claude 3.5 Sonnet               0.1886355
OpenAI o3-Google Gemini 2.0 Flash         0.3606857
```

The TukeyHSD analysis shows no statistical significance for any comparison between the models. The p-value for the comparison between Google Gemini 2.0 Flash and Claude 3.5 Sonnet is close to 0.05 and therefore i could be argued that preferring Claude over Google Gemini would still be beneficial but there is no statistical proof to that.

T-Test - Time

In the third step, we will examine each groups more closely using T-Tests.

Prompting Strategy

```
subset_data <- subset(mainData, Strategy %in% c("low_low_low", "high_high_high")) #best&wo

subset_data$Strategy <- droplevels(subset_data$Strategy)

t_test_result <- t.test(Time ~ Strategy, data = subset_data)
print(t_test_result)
```

```
    Welch Two Sample t-test

data:  Time by Strategy
t = 3.4581, df = 85.63, p-value = 0.0008497
alternative hypothesis: true difference in means between group low_low_low and group high_hig
95 percent confidence interval:
 0.0527440 0.1954041
sample estimates:
   mean in group low_low_low mean in group high_high_high
                   0.3451852                    0.2211111
```

The pairwise comparison between the two groups “low_low_low” and “high_high_high” confirms the prior analysis done. The difference in prompting strategies is statistically significant with a p-value of 0.000847.

```
subset_data <- subset(mainData, Strategy %in% c("medium_medium_medium", "high_high_high"))

subset_data$Strategy <- droplevels(subset_data$Strategy)

t_test_result <- t.test(Time ~ Strategy, data = subset_data)
print(t_test_result)
```

```
    Welch Two Sample t-test

data:  Time by Strategy
t = 2.0219, df = 87.95, p-value = 0.04623
alternative hypothesis: true difference in means between group medium_medium_medium and grou
95 percent confidence interval:
 0.00114556 0.13292851
sample estimates:
mean in group medium_medium_medium       mean in group high_high_high
                         0.2881481                          0.2211111
```

The pairwise comparison between the two groups "medium_medium_medium" and "high_high_high" shows that in isolation the difference is statistically significant (p-value of 0.04623). This is contrary to the p-value from anova. This is because anova adjusts the p-value and analyses the statistical significance across all groups and it that broader context the difference might not be statistical significant anymore. However, this is isolated comparison shows that it is still useful to opt for more detailed and optimized prompting strategies (=high_high_high).

Model

```r
subset_dataModel <- subset(mainData, Model %in% c("Google Gemini 2.0 Flash", "Claude 3.5 S

subset_dataModel$Model <- droplevels(subset_dataModel$Model)

levels(subset_dataModel$Model)
```

```
[1] "Claude 3.5 Sonnet"       "Google Gemini 2.0 Flash"
```

```r
t_test_result <- t.test(Time ~ Model, data = subset_dataModel)
print(t_test_result)
```

```
    Welch Two Sample t-test

data:  Time by Model
t = -0.38633, df = 87.469, p-value = 0.7002
alternative hypothesis: true difference in means between group Claude 3.5 Sonnet and group G
```

```
95 percent confidence interval:
 -0.08647675  0.05832860
sample estimates:
      mean in group Claude 3.5 Sonnet mean in group Google Gemini 2.0 Flash
                            0.2588889                              0.2729630
```

The pairwise comparison between the two most distinct models "Google Gemini 2.0 Flash" and "Claude 3.5 Sonnet"confirms the prior analysis done. The difference between models (comparing best to worst according to the average values) is statistically insignificant with a p-value of 0.7002.

Tobit Regression - Time

For the last step, we will use Tobit Regression to gain more insight into how each strategy influences the generation if unit-tests.

In the experimental set-up there was an upper time barrier of 30 minutes. Tobit regression handles this barrier better than a simple linear regression. Hence, it was used instead (McDonald and Moffitt, 1980).

Prompting Strategy

```r
# 0.5 is the max amount of time I spent on trying to create unit-tests using GitHub Copilo
model_tobit <- tobit(Time ~ Strategy, left = 0, right = 0.5, data = mainData)
alias(model_tobit)
```

```
list()
```

```r
summary(model_tobit)
```

```
Call:
tobit(formula = Time ~ Strategy, left = 0, right = 0.5, data = mainData)

Observations:
         Total  Left-censored     Uncensored Right-censored
           135              0             90             45

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept)                    0.43696    0.03958  11.041  < 2e-16 ***
Strategymedium_medium_medium -0.11281    0.05283  -2.135 0.032736 *
Strategyhigh_high_high       -0.19489    0.05268  -3.699 0.000216 ***
Log(scale)                   -1.45438    0.08208 -17.719  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Scale: 0.2335

Gaussian distribution
Number of Newton-Raphson Iterations: 3
Log-likelihood: -37.6 on 4 Df
Wald-statistic: 13.71 on 2 Df, p-value: 0.0010549
```

Finally, the tobit regression confirms prior analysis regarding prompting strategies. With a p-value of only 0.0002 the change from the "low"- prompting strategy to the "high"-prompting strategy is statistically significant. Also a change from the "low"- prompting strategy to the "medium"-prompting strategy is statistically significant as well. However, comparing the p-values of 0.0002 and 0.03 (which is on the edge of statistical significance) highlights the strength of the "high"-prompting strategy.

Model

```r
model_tobit <- tobit(Time ~ Model, left = 0, right = 0.5, data = mainData)
summary(model_tobit)
```

```
Call:
tobit(formula = Time ~ Model, left = 0, right = 0.5, data = mainData)

Observations:
         Total  Left-censored     Uncensored Right-censored
           135              0             90             45

Coefficients:
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                   0.29487    0.03744   7.876 3.38e-15 ***
ModelGoogle Gemini 2.0 Flash  0.02494    0.05290   0.471   0.6374
ModelOpenAI o3                0.09054    0.05343   1.694   0.0902 .
Log(scale)                   -1.41795    0.08231 -17.228  < 2e-16 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Scale: 0.2422

Gaussian distribution
Number of Newton-Raphson Iterations: 3
Log-likelihood: -42.96 on 4 Df
Wald-statistic: 3.042 on 2 Df, p-value: 0.21848
```

Finally, the tobit regression confirms prior analysis regarding different models. With a overall p-value of 0.21848 the choice of model does not statistically significant impact cycle time. Also the coefficients regarding direct model comparisons do not show any statistical significance.

Evaluation - Time

The various statistical tests showed that the prompting strategy as predictor has (statistical significant) influence on the cycle time. In practice, this means that using GitHub Copilot can and should be used to artificially create unit-tests in regards to cycle time.

The model however, does not have any statistical significance on cycle time according to the statistical methods conducted. This makes the analysis of correct prompting strategies even more valuable as they seem to have similar effects across different models.

Coverage Analysis

The second part of this prompt engineering handbook will cover the outcome variable "coverage" and will perform the mentioned statistical tests to gain insight into the possible relationship between coverage and prompting strategy and/or model. At first, there will be a correlation analysis for a broad overview. Then followed by anova to gain a more detailed look into the effects across groups. Subsequently, there will be T-Tests to examine each groups more closely. Last but not least, there will be the tobit regression analysis to gain more insight into how each strategy influences the generation if unit-tests.

Correlation - Coverage

In the first step, we will have a look at Correlation as a first exploratory phase. This should provide insights into a general direction and if it is even worth to explore further.

Prompting Strategy

```r
cor_test_result <- cor.test(mainData$`Coverage%`, mainData$StrategyNumeric, method = "pear
print(cor_test_result)
```

```
    Pearson's product-moment correlation

data:  mainData$`Coverage%` and mainData$StrategyNumeric
t = 4.7819, df = 133, p-value = 4.539e-06
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.2288775 0.5184302
sample estimates:
     cor
0.383023
```

The correlation analysis confirms that there is a moderate relationship between coverage and prompting strategy. A correlation coefficient of 0,38 suggests a strong moderate positive relationship (Akoglu, 2018). Additionally, the p-value = 0.0000046 as well as the 95% confidence interval confirm this relationship further.

ANOVA - Coverage

Now, we will use anova to gain a more detailed look into the effects across groups.

Prompting Strategy

```r
aov_coverageStrategy <- aov(`Coverage%` ~ Strategy, data = mainData)
summary(aov_coverageStrategy)
```

```
             Df Sum Sq Mean Sq F value   Pr(>F)
Strategy      2  3.583  1.7915   11.78 1.96e-05 ***
Residuals   132 20.078  0.1521
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The anova model shows that the prompting strategy significantly influences the test coverage with a p-value of 0.00002.

TukeyHSD

The following code performs a post-hoc analysis on the anova model using TukeyHSD in order to see the importance of each level for the predictor strategy.

```
TukeyHSD(aov_coverageStrategy, "Strategy")
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = `Coverage%` ~ Strategy, data = mainData)

$Strategy
                                          diff         lwr       upr     p adj
medium_medium_medium-low_low_low     0.2574444  0.06254610 0.4523428 0.0060376
high_high_high-low_low_low           0.3927778  0.19787943 0.5876761 0.0000139
high_high_high-medium_medium_medium  0.1353333 -0.05956501 0.3302317 0.2301479
```

The TukeyHSD method clearly shows that switching from the "low" to either "medium" or "high"-strategy significantly influences the test coverage (=higher test coverage). Especially the change from "low" to "high" stands out with a very small p-value of 0.0000139.s

The switch from "medium" to "high" however, seems to be statistically insignificant according to TukeyHSD. This will be further analysed using T-Tests. Still, the anova model recommends switching from "low" to "high" instead of "low" to "medium" as the corresponding p-value is much smaller.

Model

```
aov_coverageModel <- aov(`Coverage%` ~ Model, data = mainData)
summary(aov_coverageModel)
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
Model         2  0.109 0.05459   0.306  0.737
Residuals   132 23.551 0.17842
```

The anova model shows that the choice of model does not significantly influence the test coverage with a p-value of 0.737.

T-Test - Coverage

In the third step, we will examine each groups more closely using T-Tests.

Prompting Strategy

```r
subset_data <- subset(mainData, Strategy %in% c("low_low_low", "high_high_high")) #best&wo

subset_data$Strategy <- droplevels(subset_data$Strategy)

t_test_result <- t.test(`Coverage%` ~ Strategy, data = subset_data)
print(t_test_result)
```

```
    Welch Two Sample t-test

data:  Coverage% by Strategy
t = -4.8471, df = 87.649, p-value = 5.375e-06
alternative hypothesis: true difference in means between group low_low_low and group high_hig
95 percent confidence interval:
 -0.5538242 -0.2317314
sample estimates:
   mean in group low_low_low mean in group high_high_high
                   0.3763333                    0.7691111
```

As already shown by the anova model, the T-Test confirms that the “high”-strategy significantly influences the test coverage compared to the “low”-strategy.

```r
subset_data <- subset(mainData, Strategy %in% c("medium_medium_medium", "high_high_high"))

subset_data$Strategy <- droplevels(subset_data$Strategy)

t_test_result <- t.test(`Coverage%` ~ Strategy, data = subset_data)
print(t_test_result)
```

```
    Welch Two Sample t-test

data:  Coverage% by Strategy
t = -1.6597, df = 87.508, p-value = 0.1006
alternative hypothesis: true difference in means between group medium_medium_medium and grou
```

```
95 percent confidence interval:
 -0.29739489  0.02672823
sample estimates:
mean in group medium_medium_medium       mean in group high_high_high
                         0.6337778                          0.7691111
```

As already shown by the anova model, the T-Test confirms that the "high"-strategy does not significantly influence test coverage compared to the "medium"-strategy. However, the average test coverage for "medium" is only 63%, while the average test coverage for "high" is 77%. In the test project at POI the current test coverage is just about 80%. Hence, in practice - while not statistically significant - the "high"-strategy is clearly recommendable if you try to reach existing levels of test coverage.

Model

```r
t_test_result <- t.test(`Coverage%` ~ Model, data = subset_dataModel)
print(t_test_result)
```

```
    Welch Two Sample t-test

data:  Coverage% by Model
t = 0.81422, df = 87.913, p-value = 0.4177
alternative hypothesis: true difference in means between group Claude 3.5 Sonnet and group G
95 percent confidence interval:
 -0.1000535  0.2389423
sample estimates:
      mean in group Claude 3.5 Sonnet mean in group Google Gemini 2.0 Flash
                            0.6262222                           0.5567778
```

The T-Test regarding the choice of model and the influence on test coverage confirms prior analysis and the p-value of 0.4177 clearly labels the choice of model statistically insignificant.

Linear Regression - Coverage

For the last step, we will use a LM-Regression to gain more insight into how each strategy influences the generation of unit-tests.

Prompting Strategy

```r
model_lmCoverageStrategy <- lm(`Coverage%` ~ Strategy, data = mainData)
summary(model_lmCoverageStrategy)
```

```
Call:
lm(formula = `Coverage%` ~ Strategy, data = mainData)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7691 -0.3763  0.1662  0.2662  0.6237

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                   0.37633    0.05814   6.473 1.73e-09 ***
Strategymedium_medium_medium  0.25744    0.08222   3.131  0.00214 **
Strategyhigh_high_high        0.39278    0.08222   4.777 4.66e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.39 on 132 degrees of freedom
Multiple R-squared:  0.1514,    Adjusted R-squared:  0.1386
F-statistic: 11.78 on 2 and 132 DF,  p-value: 1.965e-05
```

Finally, the linear model confirms prior analysis regarding different prompting strategies. Both the "medium"-strategy (p-value of 0.00214) as well as the "high"-strategy (p-value of 0.000009) are statistically significant. The F-statistic also shows a p-value of 0.00002 further emphasizing the statistical significance. Showing that a better prompting strategy yields better test coverage. Especially the resulting p-value of the "high"-strategy makes it really clear.

Model

```r
model_lmCoverageModel <- lm(`Coverage%` ~ Model, data = mainData)
summary(model_lmCoverageModel)
```

```
Call:
lm(formula = `Coverage%` ~ Model, data = mainData)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-0.6262 -0.5568  0.1738  0.3538  0.4432

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                   0.62622    0.06297   9.945   <2e-16 ***
ModelGoogle Gemini 2.0 Flash -0.06944    0.08905  -0.780    0.437
ModelOpenAI o3               -0.03000    0.08905  -0.337    0.737
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4224 on 132 degrees of freedom
Multiple R-squared:  0.004614,  Adjusted R-squared:  -0.01047
F-statistic: 0.306 on 2 and 132 DF,  p-value: 0.7369
```

Finally, the linear model confirms prior analysis regarding different models. With an overall p-value of 0.7369 the choice of model does not statistically significant impact cycle time at all.

Evaluation - Coverage

Overall, while not as clear as for cycle time, the prompting strategy impacts the test coverage. Generally speaking better and more advanced prompting strategies (=high_high_high) result in better test coverage and hence, should be opted for than "lower" or "medium"-strategies.

'Passed' Analysis

The following chapter analyses the effect that prompting strategy and/or model have on the outcome variable "Passed". This is important because part of the experimental setup was a hard cut at 30 Minutes per prompting strategy per file for which unit-tests should be created.

If GitHub Copilot was able to create running tests within 30 Minutes that satisfied the minimal requirements regarding test coverage (at least 60% coverage, edge case tests, negative tests, parametrized tests), the passed variable was set to 1 (=True) and if not, it was set to 0 (=False). At the beginning of the thesis, this was not planned. However, due to the complexity of certain code samples and sometimes hallucinations of the models, not for all selected files/methods Github Copilot could succesfully create proper unit-tests within 30 minutes which is why the "Passed"-variable was introduced.

The following chapter is a small analysis covering that outcome variable.

Logistic Regression - Passed

Here, Logistic Regression will be used as it is preferable for binary outcome variables (Jurafsky and Martin, 2025).

Prompting Strategy

```r
model_passed <- glm(Passed ~ Strategy, data = mainData, family = binomial)
summary(model_passed)
```

```
Call:
glm(formula = Passed ~ Strategy, family = binomial, data = mainData)

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                -0.04445    0.29822  -0.149  0.88151
Strategymedium_medium_medium  0.94524    0.44397   2.129  0.03325 *
Strategyhigh_high_high        1.57593    0.49088   3.210  0.00133 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 170.44  on 134  degrees of freedom
Residual deviance: 158.59  on 132  degrees of freedom
AIC: 164.59

Number of Fisher Scoring iterations: 4
```

The logistic regression shows that the strategy is an important predictor for "Passed" equaling 1. Especially the "high"-Strategy has an significant impact on that outcome variable with a p-value of 0.00133. This again proves that good prompting strategies are necessary to a more efficient use of GitHub Copilot and AI tools in general.

Model

```r
model_passed <- glm(Passed ~ Model, data = mainData, family = binomial)
summary(model_passed)
```

```
Call:
glm(formula = Passed ~ Model, family = binomial, data = mainData)

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                   0.9008     0.3289   2.739  0.00617 **
ModelGoogle Gemini 2.0 Flash -0.1059     0.4603  -0.230  0.81811
ModelOpenAI o3               -0.4018     0.4502  -0.892  0.37217
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 170.44  on 134  degrees of freedom
Residual deviance: 169.57  on 132  degrees of freedom
AIC: 175.57

Number of Fisher Scoring iterations: 4
```

Also regarding the variable "Passed", the choice of model seems to be a statistically insignificant predictor. This further emphasizes the importance of correct prompting strategies.

Grouped Analysis

The following chapter will give insight into the statistical significance of strategy and/or model when you look at both outcome variables at the same time which is important in practice.

Manova - Time & Coverage

Here, Manova (Multivariate Anova) evaluates whether the joint distribution for Time and Coverage differs across the levels of the Model factor.

Prompting Strategy

```r
manova_modelStrategy <- manova(cbind(Time, `Coverage%`) ~ Strategy, data = mainData)
summary(manova_modelStrategy)
```

```
           Df  Pillai approx F num Df den Df    Pr(>F)
Strategy    2 0.16897   6.0907      4    264 0.0001058 ***
Residuals 132
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The manova model shows that the strategy has a statistically significant impact on both outcome variables at the same time. Meaning, a good prompting strategy leads to an improved relation of coverage and time.

Model

```r
manova_modelModel <- manova(cbind(Time, `Coverage%`) ~ Model, data = mainData)
summary(manova_modelModel)
```

```
           Df  Pillai approx F num Df den Df  Pr(>F)
Model       2 0.08698   3.0008      4    264 0.01904 *
Residuals 132
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The manova model shows that the predictor model has a statistically significant impact on both outcome variables at the same time. Meaning, the choice of model most likely does improve the relation of coverage and time.

This might seem contrary to the analysis done in prior chapters and the choice of models might actually play a **minor** role. However, the manova model also shows that the prompting strategy as a predictor with a Pillai's trace of 0.16897, an approximate F-value of 6.0907 and a p-value of 0.00011 compared to the model with a Pillai's trace of 0.087, an approximate F-value of 3 and a p-value of 0.01904 is significantly more important than the model. This indicates that the prompting strategy is far more important regarding the combination of cycle time and test coverage.

In combination with the isolated statistical tests, the manova model deepens the conclusion that good and advanced prompting strategies are the key to an efficient use of AI tools like GitHub Copilot.

Manova - Time & Passed

Here, Manova (Multivariate Anova) evaluates whether the combination of Time and "Passed" differs across the different prompting strategies.

Prompting Strategy

```
manova_modelStrategy <- manova(cbind(Time, Passed) ~ Strategy, data = mainData)
summary(manova_modelStrategy)
```

```
           Df  Pillai approx F num Df den Df   Pr(>F)
Strategy    2 0.10791   3.7641      4    264 0.005371 **
Residuals 132
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As the manova model shows the prompting strategy has an statistical significant impact on successful attempts. In other words, a good prompting strategy helps upholding minimum requirements (and beyond).

Model

```
manova_modelStrategy <- manova(cbind(Time, Passed) ~ Model, data = mainData)
summary(manova_modelStrategy)
```

```
           Df   Pillai approx F num Df den Df Pr(>F)
Model       2 0.033232   1.1152      4    264 0.3497
Residuals 132
```

According to this analysis, the choice of model does not impact if the attempts during the experimental phase were successful or not.

Other findings/optimizations

This chapter will give insight into findings and leanings from the experimental phase. This has little do with in particular with either prompting strategies or models but applies to the general usage of GitHub Copilot to work more efficiently regardless of the concrete model or prompting strategy used.

Precise commands

In many cases - especially with "lower" "medium" prompting strategies but also with higher prompting strategies - the model suffers from hallucination and tries to change the original files that needed to be tested to make the tests work. A simple "Do not change original methods" completely negates that time wasting side-effect and makes the usage a lot more efficient and also less frustrating.

Additionally, it helps a lot to mention the minimum requirement regarding test coverage you want the tests to reach (let's say 80%). It is not certain that it will help but often it felt like that it pushes the model in the directions at least.

Evaluation

Finally, there will be a last discussion regarding all insights gained. This will include a short summary, followed by recommendations based on the findings and lastly, an outlook for further research.

This prompt engineering handbook takes a look and detailly analysed the effects that different prompting strategies and models have on test coverage and cycle time (the less time needed to artificially create good unit-tests, the more the cycle time improves). Using statistical testing each effect and interaction was carefully evaluated and put into perspective. The effects of different model choices were also examined in order to prove that a good prompting strategy makes the difference and not "external" factors like model quality is. Even when there might be differences in models, the prompt engineering handbook showed clearly that having a good and sophisticated prompting strategy is far more (statistically speaking) important.

Additionally, during the course of this practice-oriented thesis, it showed that the inclusion of examples, a specific test coverage target as well commands like "Do not change the original methods" greatly help the model to stay focused on it's task, avoid hallucinations and "side-quests" and deliver higher quality unit-tests.

To summarize, in the case of the "high"-strategy this consistent of the following things:

1. Degree of detail of the prompt "high":

   The concrete method that needs to be tested as well as helper methods or methods where it is used. Moreover, the structure of the test will be mentioned (e.g. parameterized test). Additionally, the task within the story is mentioned as well as already existing, functioning examples. Additionally, mention the test coverage you aim at.

2. Level of context "high":

   The file with the code that needs to be tested, the test class as well as files that use the methods. Additionally, if the test should use certain fixtures, these files be added via the button "Add Files" as well to provide even more context.

3. Quality of the prompt “high”:

   “Imagine you are a highly experienced Senior (Java-)Developer with a strong background in writing unit tests. Your task is to create unit-tests for the newly added method(s). Think about how your test methods can include normal functionality, edge cases, parametrized and negative tests and then write suitable test(s). Of course, only add special test cases when they make sense. Additionally, aim for a high test coverage of xx% and do not change the original methods.”

To generalize this: State clearly and precisely what you want the model to do (=explicit instructions), add as much context as possible, specify the language and assign a role to the model (like Senior Developer). Additionally try to trigger a chain-of-thought process which has shown to greatly increase the quality of the model's responses (e.g. “Think about how your test methods can include normal functionality, edge cases, parametrized and negative tests and then write suitable test(s).” (Wei et al., 2022).

As mentioned at the beginning of this thesis, this is a vast topic and despite the efforts of this thesis, much needs still to explored. It will take a lot more and detailed analysis to truly find the “best prompting strategy”. Especially, if thinking about topics like complete automation (e.g. using the identified strategies to automatically trigger Copilot or different AI tools) , there needs to be a lot more testing and research done to enhance quality and stability of the model's output. This can be only reached by even more refined prompting techniques which are the connection between humans and AI models. This paper works more as a prototype, a concept to testify the importance of prompt engineering and works as a first guidelines what has to be considered and how the output of AI tools like GitHub Copilot can be enhanced.

To answer the secondary research questions proposed in Chapter 1 of the thesis:

Firstly, regarding reaching test coverage rates of 100%, this paper showed that there is still some instability. In some cases, Copilot runs into errors and does not manage to fulfil the minimum criteria within the 30 minutes time span. For these attempts, the coverage was set to 0% lowering the average test coverage for all files. But, and for the sake of discussion, if the average would only be calculated based off successful attempts (83% or 37 out of 45 attempts) using the ”high”-strategy, an average test coverage of 93% would be reached. This might not be 100% yet but pushes heavily into that direction. Excluding failed attempts, however, seems nonsensical as especially these cases demand further research and optimization. To conclude, the paper shows that reaching existing test coverage rates of around 80% is already feasible and that more advanced prompting strategies enhance output stability (=less failed attempts). However, Copilot cannot reach test coverage rates of 100% on average yet. This is especially due to the models running into error loops, hallucinations and most likely the complexity of certain code structures.

Furthermore, this paper shows clearly that the prompting strategy matters, and a sophisticated and clear techniques positively influences cycle time and test coverage (and in this regard generate “high quality” unit-tests as they satisfy all set constraints). A possibility to integrate

the good prompting strategies into developer workflows is the implementation of GitHub Copilot extensions that leverage the well-founded prompting strategies and make them accessible for all users. This approach was demonstrated by developing the small additional prototype during this thesis.

The research done shows that Copilot has the capabilities to save 75% (30 m cut-off in the experiment vs. 2h minimum current development time) of development time and yet maintaining existing test coverage rates (current test of 85% compared to the 82% reached by the "high"-strategy). This would mean cost savings of 9,84 €s per hour based on the Austrian IT collective bargaining agreement (WKO, 2024).

To conclude, AI tools like an efficient use of GitHub Copilot can positively affect test coverage and cycle time (cycle time used as a measurement to not only talk about performance but also financial standpoints). This becomes especially apparent, if you look at them in relation to each other.

Sources

Here all sources specifically used in this handbook are listed.

Abdi, H. and Williams, L.J. (2010) 'Tukey's honestly significant difference (HSD) test', in Salkind, N. S. (ed.) *Encyclopedia of research design.* Thousand Oaks, CA: Sage. Available at: https://personal.utdallas.edu/~herve/abdi-HSD2010-pretty.pdf (Accessed: 1 April 2025).

Akoglu, H. (2018) 'User's guide to correlation coefficients', *Turk J Emerg Med*, 18(3), pp. 91–93. Available at: https://pmc.ncbi.nlm.nih.gov/articles/PMC6107969/pdf/main.pdf (Accessed: 1 April 2025).

Huberty, C.J. and Olejnik, S. (2006) Applied MANOVA and discriminant analy-sis. 2nd edn. Hoboken, NJ: John Wiley & Sons. Available at: http://ndl.ethernet.edu.et/bitstream/123456789/27897/1/Carl% (Accessed: 27 March 2025).

Janse, R.J., Hoekstra, T., Jager, K.J., Zoccali, C., Tripepi, G., Dekker, F.W. and van Diepen, M. (2021) 'Conducting correlation analysis: important limitations and pitfalls', *Clinical Kidney Journal*, 14(11), pp. 2332–2337. Available at: https://doi.org/10.1093/ckj/sfab085 (Accessed: 1 April 2025).

Jurafsky, D. and Martin, J.H. (2025) Speech and language processing: an in-troduction to natural language processing, computational linguistics, and speech recognition with language models. 3rd edn. (online) Stanford Universi-ty. Available at: https://web.stanford.edu/~jurafsky/slp3/5.pdf (Accessed: 1 April 2025).

Kim, T.K. (2015) 'T test as a parametric statistic', Korean Journal of Anaes-thesiology, 68(6), pp. 540–546. Available at: https://doi.org/10.4097/kjae.2015.68.6.540 (Accessed: 1 April 2025).

Kutner, M.H., Nachtsheim, C.J., Neter, J. and Li, W. (2005) Applied linear sta-tistical models. 5th edn. Boston: McGraw-Hill/Irwin. Available at: https://users.stat.ufl.edu/~winner/sta4211/ALSM_5Ed_K (Accessed: 1 April 2025).

McDonald, J.F. and Moffitt, R.A. (1980) 'The uses of Tobit analysis', The Re-view of Economics and Statistics, 62(2), pp. 318–321. Available at: http://www.jstor.org/stable/1924766 (Accessed: 1 April 2025).

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. and Zhou, D. (2023) Chain-of-thought prompting elicits reasoning in large language models. Available at: https://doi.org/10.48550/arXiv.2201.11903 (Accessed: 20 January 2025).

Wirtschaftskammer Österreich (2024) *Kollektivvertrag Informationstechnologie 2024*. Available at: https://www.wko.at/oe/kollektivvertrag/kv-informationstechnologie-2024.pdf (Accessed: 20 April 2025).