1 Identify languages by text analysis (Projekt 1)

1.1 A program that distinguishes Swedish from Norwegian

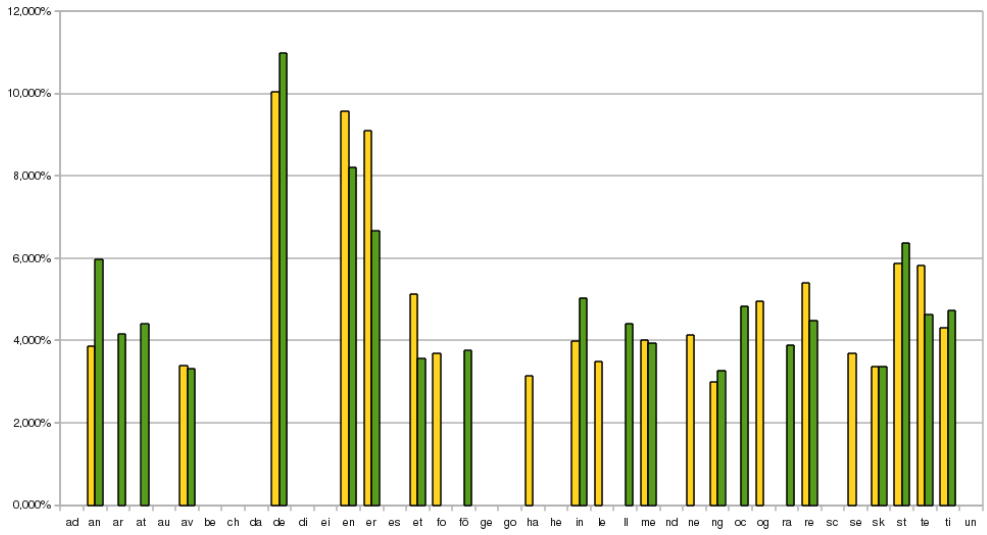
example texts As example texts to work with I have fetched ∼20MB of data from Wikipedia using a shellscript that downloads the list of the "excellent articles" in the Norwegian and Swedish Wikipedia (currently ∼200 articles each) and strips the relevant text out of the downloaded HTML-sourcecode (total word count of all wikipedia samples after processing: 2 million words)

In order to try the "extra challange" I additionally fetched example texts from "bild.de" and "faz.net" using a shellscript that downloaded the RSS-feeds of the newspaper-websites and fetched all its articles (∼40 articles fetched) and stripped the relevant text out of the HTML-sourcecode.

All the sample texts were stored as several text-files in a directory for each language ("no/", "sv/") and newspaper ("bild/", "faz/") for later use (statistics!)

differential language features In order to find features that differ from language to language I wrote a script that analyzed the frequency of several things (most frequent words, most frequent characters and most frequent two-character-combinations). By looking at the data I concluded that the two-character-combinations could allow me to differenciate the languages from each other, because there was a visible difference in the order of the most frequent ones.

So I took the relative frequencies of the most frequent two-character-combinations and made a graph out of it (see Figure 1)



language	distinct character combinations	
sv	ar,at,fö,ll,oc,ra	(all green-only-bars)
no	fo,ha,le,ne,og,se	(all yellow-only-bars)

highest distinct bars: og (yellow) / oc (green)

Figure 1: relative frequencies of the most frequent two-character-combinations of Norwegian (yellow) and Swedish (green) texts

My idea how to diffenenciate the languages from each other was now to compare the relative

frequencies of the two-character-combinations which showed the highest deviation for particular languages in the graph.

So I wrote a initial R program that compared the frequency of “oc”, “og” (and “ei” for German, for the “extra challange”) in a given file.

testing Then I wrote a script that called my inital R program several times with a portion of text of different length for each language and summed up the results.

I put these results into a graph (see Figure 2), then I tried to improve my R program using the results I got out of the graph and repeated the test run...

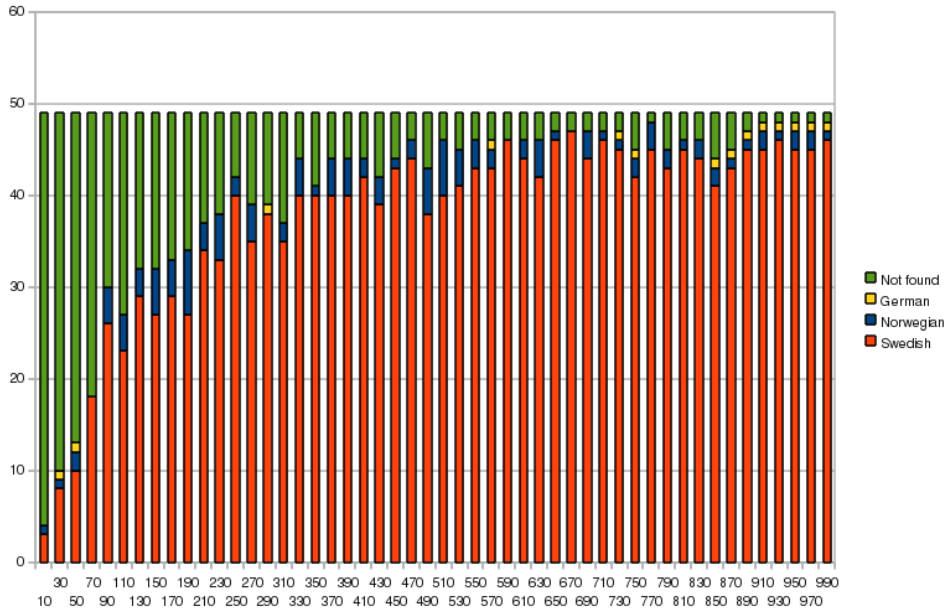


Figure 2: recognition results of the **initial** program with random Swedish (red) sample text portions with various lengths from 10 to 1000 characters

I repeated to improve and test the program until the graph showed a better recognition rate than the initial graph (see Figure 3): I adjusted the limits of the relative frequencies (i.e. how much percentage points are necessary for the relative frequency of a two-character-combination to identify a language) and I added a fallback routine which is called when the language couldn’t be identified in the first run. This routine is just like the main one, but it searches for other language-specific two-character-combinations (as you can see in Figure 1 there are more than just “og” and “oc”, in my case I’m using “ar” and “se”). This fallback routine increases the possibility for a short text to be identified, because especially in short texts there might be no occourrences of “og” and “oc” at all, and without this fallback it would simply not get identified, whereas with this fallback it has got a “second chance”.

error rates The estimated error rates are (roughly) apparent from the graph of Figure 3, but here are the values for the lengths 20, 200 and 2000 (as requested):

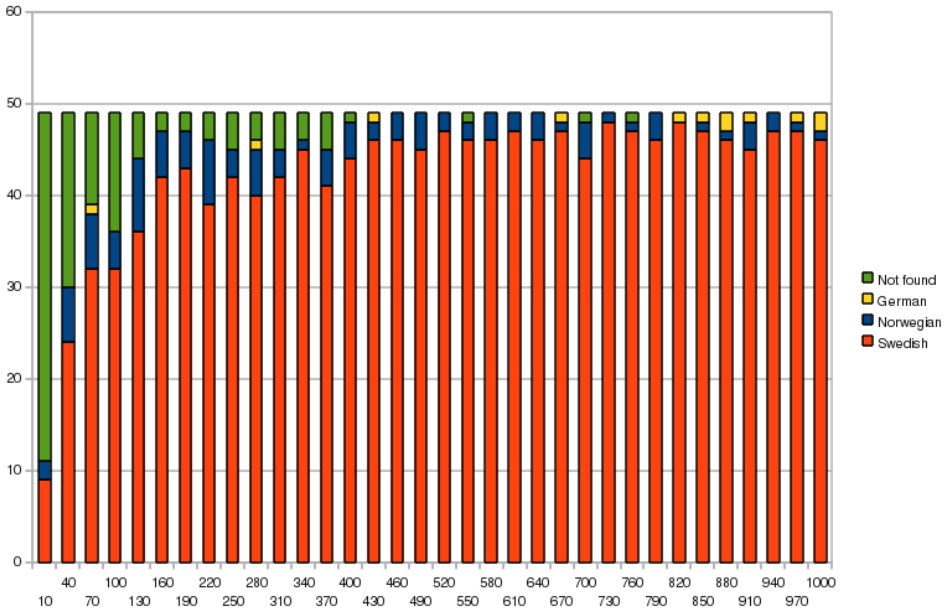


Figure 3: recognition results of the **improved** program with random Swedish (red) sample text portions with various lengths from 10 to 1000 characters

length	language	identified correctly	identified incorrectly	not identified	# tests
20	Swedish	134 (26%)	16 (3.2%)	349 (69.8%)	500
20	Norwegian	101 (20.2%)	52 (10.4%)	346 (69.2%)	500
200	Swedish	245 (81.6%)	40 (13.3%)	14 (4.6%)	300
200	Norwegian	236 (78.6%)	43 (14.3%)	20 (6.6%)	300
2000	Swedish	190 (95%)	9 (4.5%)	0 (0%)	200
2000	Norwegian	191 (95.5%)	8 (4%)	0 (0%)	200

limitations

- the alogrithm uses statistical analysis which can never detect the language for sure, you can only assume that for a certian probability (depending on factors like length, type of text, ...) the computed result is right.
- the shorter the portion of text is the lower the probability of a correct language detection is (see graph Figure 3 or table error rates)
- if the text contains large amounts of non-language-elements (like names, formulas or other data) the detection is likely to be wrong because these non-language-elements can't be stripped out and will be used for language detection, which may falsify the result.

1.2 "Extra challenge": distinguish texts taken from the FAZ from those you find in Bild

algorithm This algorithm is based on the assumption that a text with a high count of "?" and "!" is likely to be a "Bild"-text, because Bild quite often uses sensational expressions, in contrast to the FAZ, which uses a more complex and less emotional language.

error rates The error rates for distinguishing Bild from FAZ are worse than those for Swedish from Norwegian (see Figures 4 and 5). Much longer texts are needed to be able to identify either Bild or FAZ.

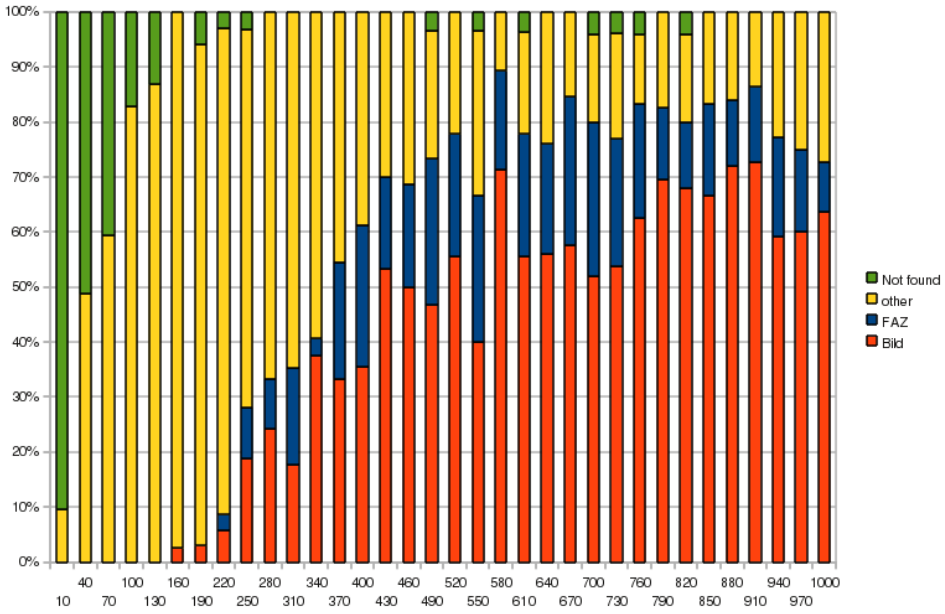


Figure 4: recognition results with random Bild (red) sample text portions with various lengths from 10 to 1000 characters

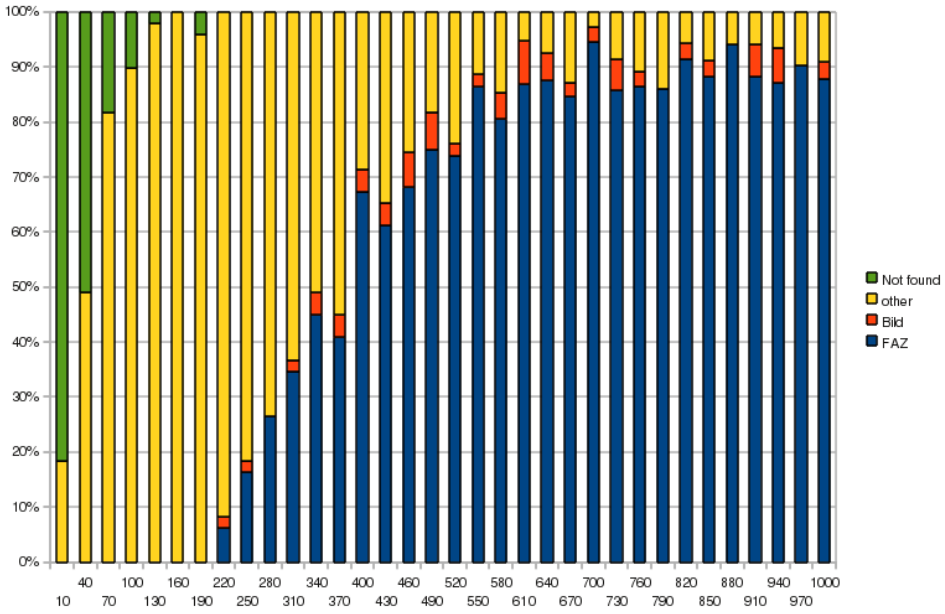


Figure 5: recognition results with random FAZ (blue) sample text portions with various lengths from 10 to 1000 characters

Listing 1: projekt1.R

```
#!/usr/bin/Rscript
# Genomik und Bioinformatik I
# Projekt 1 (Identify languages by text analysis)
# written by Andreas Loibl (enrollment no. 1524148)

source(file="readText.R")

# identifyGermanNewspaper:
##############################
#    distinguishes texts from the German newspapers FAZ and Bild by analyzing
#    the relative frequency of punctuation marks in the text. This algorithm
#    is based on the assumption a text with a high count of "?" and "!" is
#    likely to be a "Bild"-text, because Bild quite often uses sensational
#    expressions.
#
#    So this function is able to decide if the given text is taken from Bild
#    or from FAZ (or generic "Text" if the data is not distinguishable enough
    )

identifyGermanNewspaper <- function(inputText) {
    count <- numeric()
    for(i in c("!", "\\?", "\\. ", ",")) {
        tmp <- unlist(gregexpr(i,inputText))
        if(tmp[1] > 0) count[i] <- length(tmp) else count[i] <- 0
    }
    if(sum(count)>5) {
        if((count["\\?"]+count["!"])/sum(count) > 0.1) return("Bild")
        if((count["\\?"]+count["!"])/sum(count) < 0.05) return("FAZ")
```

```
28          print("Data is not distinguishable enough.")
29          return("Text")
30      } else {
31          print("No or not enough processable data found.")
32          return("Text")
33      }
34  }
35
36  # identifyLanguageOfFile:
37  ############################
38  #   distinguishes the language of a file by using text analysis on its
        content.
39  #
40  #   the algorithm counts the occurrences of several character-combinations
41  #   (like "oc", "og", "ei", "ar", "se") and estimates what language the text
42  #   is written in by comparing the relative frequencys of these character-
43  #   combinations with values.
44
45  identifyLanguageOfFile <- function(file) {
46      inputText <- readText(file)
47      count <- numeric()
48      for(i in c("og", "oc", "ei")) {
49          tmp <- unlist(gregexpr(i,inputText))
50          if(tmp[1] > 0) count[i] <- length(tmp) else count[i] <- 0
51      }
52      if(sum(count)>0) {
53          if((count["og"])/sum(count) > 0.40) return("Norwegian")
54          if((count["oc"])/sum(count) > 0.50) return("Swedish")
55          if((count["ei"])/sum(count) > 0.45) return(paste("German", "(",
                identifyGermanNewspaper(inputText), ")"))
56      }
57      count <- numeric()
58      for(i in c("ar", "se")) {
59          tmp <- unlist(gregexpr(i,inputText))
60          if(tmp[1] > 0) count[i] <- length(tmp) else count[i] <- 0
61      }
62      if(sum(count)>0) {
63          if((count["ar"])/sum(count) > 0.65) return("Swedish")
64          if((count["se"])/sum(count) > 0.65) return("Norwegian")
65          print("Data is not distinguishable enough.")
66          return(FALSE)
67      }
68      print("No processable data found.")
69      return(FALSE)
70  }
71
72  for(file in commandArgs(TRUE)) print(identifyLanguageOfFile(file))
```