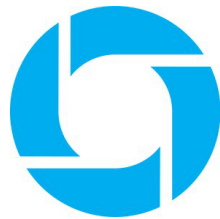


# Developing a book recommendation system

**Radboud University**



**Onderwijs in Beeld**

Vooruitstrevend leesonderwijs

Name: Harm Jacobs  
Student number: s1019253  
Supervisor at university: Arjen de Vries  
Supervisor at company: Stijn Dautzenberg  
Second examiner: Djoerd Hiemstra  
Date: 27-06-2023

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction and research question</b>                        | <b>5</b>  |
| <b>2</b> | <b>Data collection</b>   | <b>7</b>  |
| 2.1      | Titelbank  | 7         |
| 2.2      | FTP connection   | 9         |
| 2.3      | Not enough data  | 10        |
| 2.4      | Keywords   | 11        |
| 2.5      | AVI  | 16        |
| 2.6      | CLIB-niveau  | 17        |
| 2.7      | PDF-parser   | 18        |
| <b>3</b> | <b>Recommendation Algorithm</b>                                  | <b>23</b> |
| 3.1      | Content-based filtering  | 23        |
| 3.2      | Collaborative filtering  | 24        |
| 3.3      | ANNOY  | 25        |
| 3.4      | Different metric options   | 29        |
| 3.5      | Curse of dimensionality  | 31        |
| 3.6      | Limit/inside ratio   | 34        |
| 3.7      | Calculations   | 36        |
| 3.7.1    | Euclidean  | 36        |
| 3.7.2    | Manhattan distance   | 36        |
| 3.7.3    | Angular distance   | 36        |
| 3.7.4    | Dot product distance   | 37        |
| 3.7.5    | Hamming distance   | 37        |
| 3.7.6    | Conclusion   | 37        |
| <b>4</b> | <b>Implementation</b>  | <b>38</b> |
| 4.1      | Database structure   | 38        |
| 4.2      | Data preparation   | 39        |
| 4.3      | Recommendation formula   | 41        |
| <b>5</b> | <b>Results</b>   | <b>44</b> |
| <b>6</b> | <b>Future work</b>   | <b>47</b> |
| <b>7</b> | <b>Appendix</b>  | <b>48</b> |
| <b>A</b> | <b>Preprocessing data</b>  | <b>48</b> |
| <b>B</b> | <b>Code for generating random points in different dimensions</b> | <b>53</b> |
| B.1      | Random points 1D   | 53        |
| B.2      | Random points 2D   | 54        |
| B.3      | Random points 3D   | 55        |
| <b>C</b> | <b>Euclidean vs Angular vs Manhattan vs Dot product distance</b> | <b>56</b> |

|          |                                    |           |
|----------|------------------------------------|-----------|
| <b>D</b> | <b>Code for limit/inside ratio</b> | <b>58</b> |
| D.1      | 1D                                 | 58        |
| D.2      | 2D                                 | 59        |
| D.3      | 3D                                 | 60        |

## **Abstract**

The reading level of children in the Netherlands is decreasing. Onderwijs in Beeld is trying to turn this around and wants to make reading more enjoyable for children under the motto: "The right book for every student". In order to achieve this, a book recommendation system is needed. This thesis shows how Onderwijs in Beeld collected the necessary information about the books, the underlying techniques behind current recommendation systems and how the collected information is utilized in the final implementation of the recommendation system. The thesis concludes with a few recommendations from the implemented book recommendation system and an outlook into the future on how the recommendation system can be further enhanced in followup work.

## Preface

This bachelor thesis is part of the Computing Science bachelor at Radboud University in Nijmegen. During the thesis project I acquired new programming skills which may prove useful in the future. In addition, I gained knowledge about complications that occur in high dimensional spaces.

I would like to thank Stijn Dautzenberg and Tom Buunk for giving me the opportunity to do my bachelor thesis at their company Onderwijs in Beeld as well as supervising me. I would also like to thank Arjen de Vries for supervising me during this thesis and assisting me in finding the best solutions to some problems that we encountered.

# 1 Introduction and research question

Every 3 years, the scholastic performance such as reading level is measured and compared among countries [1]. The results are concerning, especially for the Netherlands:

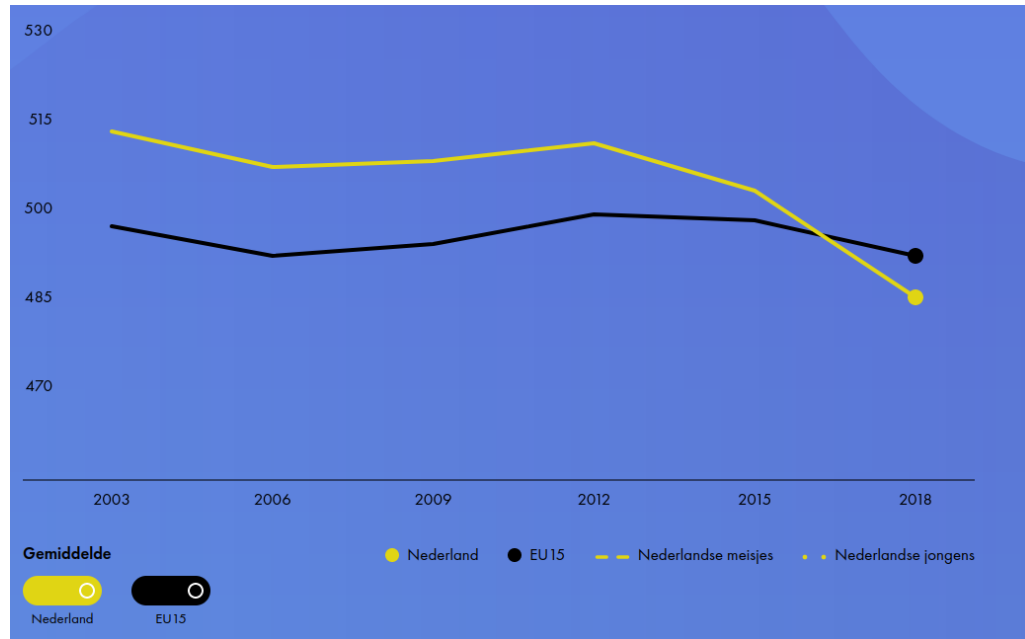


Figure 1: Reading level results from PISA

Source: [https://www.pisa-nederland.nl/resultaten2018/#sect\\_leesvaardigheid](https://www.pisa-nederland.nl/resultaten2018/#sect_leesvaardigheid)

As can be observed, the reading level among Dutch students is decreasing rapidly. The two main reasons for this are[2]:

1. Students are not reading books that fit their reading level, personality or interests.
2. Teachers have a limited view on the reading level and progress of their students and have little involvement in the process.

In order to turn this around, Stijn Dautzenberg and Tom Buunk founded Onderwijs in Beeld with one goal: improve the reading pleasure of 100 000 students at the elementary schools before 2025. Together they made an application with a wide range of features. The most important feature, also the most relevant for this thesis, is that Onderwijs in Beeld has hundreds of thousands of books in their database with information like author, recommended age, description, genre etc. Utilizing the book information from their database, the app provides schools with the capability to scan their libraries to create

a digital copy, which results in a clear inventory of the books available in the school. This enables children to borrow books so teachers can observe which student is currently reading which book, as well as previously read books. When students borrow books using the application, they can enter the progress that they made resulting in digital logs that can be monitored by teachers to gain deeper insights of their students reading activities.

Once this part of the application was functioning properly, Onderwijs in Beeld proceeded to the subsequent phase: "The perfect book for every student", which is the focus of the thesis. Hence, the main research question:

"How to develop a book recommendation system in order to ensure that the system provides users with recommendations of their level and interests?"

Since this is a broad question that lacks a concise answer, the main question is divided into the following subquestions:

1. What methods are available to acquire information about books?
2. What are the underlying techniques behind current recommendation systems?
3. How can the collected information be utilized for the implementation of the recommendation system?

The first part of this thesis will revolve around how Onderwijs in Beeld managed to obtain the data about the books in their database and how it stays up-to-date. It also provides a more in-depth explanation about the reasons and methods behind the collection of this information as well as the most relevant information for choosing a certain book.

The second part explains different techniques that are often used for recommendation systems alongside some examples. It also analyzes the workings of the ANNOY algorithm, which is one of the algorithms often utilized in recommendation systems and is also the algorithm implemented in the final version of the recommendation system in the Onderwijs in Beeld app.

The final part demonstrates how the collected information is prepared to make it usable in the recommendation system alongside the final implementation itself. In addition, it also discusses future work possibilities that can further enhance the system, but would require data from users in order to be an improvement.

## 2 Data collection

Before a book recommendation system can be developed, it is important to possess high quality and quantity data about books. The primary meta-data for a book is its title and an image of the book cover, since these things must be displayed in the app. However, in order to develop a book recommendation system a significantly larger amount of data is required, especially since Onderwijs in Beeld is specialised in primary schools. In primary schools, children have just started reading, which results in the reading level of a book being critical information. After all, books for children in group 8 should not be recommended to children in group 3 because they just learned how to read and probably would not understand many words. In order to acquire all this data, a lot of different methods were utilized, of which the most important one is Titelbank.

### 2.1 Titelbank

Titelbank is a database that contains all titles of Dutch books published from 1970 [3]. On the site of Titelbank [4], a large quantity of data is available. One can for example check if the title one wishes to name their book already exists, which authors authored which books, the books ISBN (International Standard Book Number), which is a unique identifier for books and is typically written as ISBN-13 due to its 13-digit format, and various other variables.

The Titelbank also makes it possible to request a file that contains all the information they keep about the books in their database. Onderwijs in Beeld also requested such a file as the foundation of their own database. The requested file contained information of approximately 300,000 books and for most of these books an image of the cover and a PDF file of the first few pages were also included:










|   |                               |          |              |                      |
|---|-------------------------------|----------|--------------|----------------------|
|  | 9789464480665_DVB.pdf         | 327,5 kB | PDF docum... | 18 april 2022, 22:01 |
|  | 9789464481969_DVB.pdf         | 270,7 kB | PDF docum... | 18 april 2022, 22:01 |
|  | TB30_update_2022-04-18_618... | 268,2 kB | unknown      | 18 april 2022, 22:01 |
|  | 9789401464413_DVB.pdf         | 195,5 kB | PDF docum... | 18 april 2022, 22:01 |
|  | 9789020549133_VRK.jpg         | 186,1 kB | JPEG image   | 18 april 2022, 22:01 |
|  | 9789464182392_DVB.pdf         | 163,7 kB | PDF docum... | 18 april 2022, 22:01 |
|  | 9789403651415_DVB.pdf         | 159,9 kB | PDF docum... | 18 april 2022, 22:01 |
|  | 9789491049101_VRK.jpg         | 157,1 kB | JPEG image   | 18 april 2022, 22:01 |
|  | 9789464356793_VRK.jpg         | 118,1 kB | JPEG image   | 18 april 2022, 22:01 |

Figure 2: Layout of the Titelbank file



The file that is highlighted contains all the necessary information about the books in an ONIX file, which is a file in standardised XML format for book metadata:

```

1 <?xml version="1.0"?><ONIXMessage release="3.0" xmlns="http://ns.editeur.org/onix/3.0/
reference"><Header><Sender><SenderIdentifier><SenderIDType>10</SenderIDType><IDValue>8894126</IDValue></
SenderIdentifier><SenderName>Titelbank</SenderName></Sender><MessageNumber>6183</
MessageNumber><SentDateTime>20220418T2200</SentDateTime></Header><Product><RecordReference>9789074123068</
RecordReference><NotificationType>04</NotificationType><RecordSourceType>01</
RecordSourceType><ProductIdentifier><ProductIDType>03</ProductIDType><IDValue>9789074123068</IDValue></
ProductIdentifier><DescriptiveDetail><ProductComposition>00</ProductComposition><ProductForm>BD</
ProductForm><NoCollection></NoCollection><TitleDetail><TitleType>01</
TitleType><TitleElement><TitleElementLevel>01</TitleElementLevel><TitleText>Handleiding Inspiratiespel</
TitleText></TitleElement></TitleDetail><Contributor><SequenceNumber>1</SequenceNumber><ContributorRole>A01</
ContributorRole><PersonName>P. Gerrickens</PersonName><NamesBeforeKey>P.</
NamesBeforeKey><KeyNames>Gerrickens</KeyNames></Contributor><Contributor><SequenceNumber>2</
SequenceNumber><ContributorRole>A01</ContributorRole><PersonName>M. Verstege</PersonName><NamesBeforeKey>M.</
NamesBeforeKey><KeyNames>Verstege</KeyNames></Contributor><EditionNumber>1</
EditionNumber><Language><LanguageRole>01</LanguageRole><LanguageCode>dut</LanguageCode></Language><Extent>
2 <ExtentType>00</ExtentType>
3 <ExtentValue>87</ExtentValue>
4 <ExtentUnit>03</ExtentUnit>
5 </Extent>
6 <Illustrated>01</Illustrated><Subject><MainSubject></MainSubject><SubjectSchemeIdentifier>32</
SubjectSchemeIdentifier><SubjectCode>780</SubjectCode></Subject></
DescriptiveDetail><PublishingDetail><Publisher><PublishingRole>01</
PublishingRole><PublisherIdentifier><PublisherIDType>10</PublisherIDType><IDValue>8300948</IDValue></
PublisherIdentifier><PublisherName>Gerrickens, Uitgeverij</PublisherName></Publisher><PublishingStatus>04</
PublishingStatus><PublishingDate><PublishingDateRole>01</PublishingDateRole><DateFormat>00</
DateFormat><Date>20020403</Date></PublishingDate></
PublishingDetail><RelatedMaterial><RelatedWork><WorkRelationCode>01</
WorkRelationCode><WorkIdentifier><WorkIDType>01</WorkIDType><IDTypeName>NSTC</IDTypeName><IDValue>500718687</
IDValue></WorkIdentifier></RelatedWork></
RelatedMaterial><ProductSupply><SupplyDetail><Supplier><SupplierRole>00</
SupplierRole><SupplierName>TitelBank</SupplierName></Supplier><ProductAvailability>20</
ProductAvailability><Price><PriceType>02</PriceType><PriceAmount>45</PriceAmount><CurrencyCode>EUR</
CurrencyCode></Price></SupplyDetail></ProductSupply></Product><Product><RecordReference>9789461534835</

```

Figure 3: ONIX file

The interpretation of all tags can be found online<sup>1</sup>. All information about one book is between the `< Product >` tags and the data available for most books are:

1. ISBN-13
2. Title of the book
3. Authors, illustrators etc.
4. Language
5. Publisher
6. Published date
7. Price

This ONIX file is then parsed and the data is stored in the Onderwijs in Beeld database. This data is a great initial step, but there are always new books being published so the database requires continues updates. The Titelbank provides this possibility using an FTP server.

<sup>1</sup><https://ns.editeur.org/onix/en>

## 2.2 FTP connection

An FTP server, file transfer protocol server, is a way to transfer files between computers over a network [5]. The Titelbank has its own FTP server where a new file with updates is published daily. Utilizing the ftplib library in python [6], Onderwijs in Beeld thus updates its database as follows:

```
1 import ftplib
2
3 ftp = ftplib.FTP()
4 ftp.connect(host, port)
5 ftp.login(username, password)
6 files = ftp.nlst()
7 for file in files:
8     # Do something with the files
9
10 ftp.quit()
```

## 2.3 Not enough data

Having done all of the above, a large amount of crucial information about the books is still missing. As already mentioned before, children that just started reading should not be recommended books for advanced readers. Also, since Onderwijs in Beeld is specialised in primary schools, children should not be recommended adult books that contain horrifying or violent content. As a result, variables like AVI, which is used for measuring the reading skill level of students discussed in subsection 2.5, and age are important variables that should be available for most books. Another important variable to look at are keywords. If a student prefers reading books about cars, the system should recommend other books about cars. This requires a way to link certain keywords to books, which is discussed in section 2.4. The type of the book is yet another important variable. In primary schools, there is a wide selection of books that are for the teacher to read to the class or books that contain mainly pictures and almost no text. There are also books that contain only information and that have no story. Therefore, it is necessary that all of these different book types are taken into consideration when recommending books.

In order to obtain all these extra variables, scrapers were used. They are implemented in python using the requests [7] and BeautifulSoup [8] libraries:

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 try:
5     page = requests.get(url, timeout=10)
6 except requests.exceptions.ReadTimeout:
7     print("Timeout reached")
8
9 soup = BeautifulSoup(page.content, 'lxml')
```

The required information on the page can then be located using the `find_all()` method, which takes as argument the name of the tag one wishes to find. The data from Titelbank combined with that from the scrapers results in a database containing all the necessary information for the recommendation system.

## 2.4 Keywords

In order to match books with the same topics, a way to link keywords to books is required. The objective is to match books about football with other books about football. In addition, it is important to consider that children who read books about football would rather read a book about another sport than a book about art. To obtain the desired result, two trees were constructed.

The first data structure, `tag_word_dict`, is a dictionary that contains keywords as tags and as value another dictionary with terms as keys and as values the relevance that this word contributes to the keyword. These relevances were initialised with random values and adjusted by trial and error after the final implementation in order to achieve the best results.

```
1 TAG_KEYWORD_DICT = {
2     "hockey": {
3         "hockey": 1.4
4     },
5     "basketbal": {
6         "basketbal": 1.4
7     },
8     "skiën": {
9         "ski ": 1.3,
10        "skiën": 1.3
11    },
12    "golf": {
13        "golf ": 1.3,
14        "golfen ": 1.3,
15        "golfbaan": 1.1
16    }
17 }
```

Listing 1: Keywords tree:

Note that certain terms have spaces in front of them and others not. This is important, because some keywords would otherwise be triggered while they should not. Looking at the keyword “skiën”, Dutch for skiing, it can be seen that this keyword is triggered by the terms “ski ” and “skiën”. If the first term would be “ski” instead of “ski ” and the description of a book would be: “The hacker has insane computer skills and tries to hack the government”, the word “ski” would be triggered and the book gets assigned the keyword “skiën” which is not what this book is about. This approach of utilizing terms rather than words alone also allows to trigger certain keywords based on multiple words. The keyword “AI” for example should be triggered by either “AI ” or “artificial intelligence”, but not by “artificial” or “intelligence” alone.

The second data structure, `tag_map_interests`, is a dictionary to take into account that a single keyword can trigger other keywords. “skiing” can for example trigger both sport and holiday and sport can trigger hobbies again. The result is a tree where each keyword can have multiple sub-trees and keywords that trigger this keyword. whenever a certain keyword is triggered, all keywords higher up in the tree also get triggered with the relevance being multiplied by 0.5 each step higher up in the tree.

```

1 TAG_MAP_INTERESTS = {
2   'hobbies': {
3     'sport': {
4       'teamsport': {
5         'voetbal': 1.5,
6         'hockey': 1.5,
7         'basketbal': 1.5
8       },
9       'wintersport': {
10        'skiën': 1.5
11      },
12      'solo-sport': {
13        'darten': 1.1,
14        'schaken': 1.1,
15        'golf': 1.3
16      }
17    }
18  }
19 }

```

Listing 2: Keywords tree:

To associate these keywords with books, the description of the book is utilized. To ensure that the term “golf” is correctly triggered if it is next to punctuation marks, certain punctuation marks like point, comma, question mark and exclamation mark are replaced with a space. For all keywords in `tag_word_dict` and for all trigger words per keyword the following algorithm is applied:

- Count the number of occurrences of the trigger word.
- For each occurrence, add the associated relevance found in `tag_word_dict` to the total relevance sum per keyword.
- If this sum is greater than 0, save the keyword and its associated total relevance estimate to a dictionary.

The result is a dictionary with keywords as keys and a relevance score estimate as value. A recursive function is called on this dictionary to walk the `tag_map_interests` tree that keeps track of the total accumulated relevance per keyword. If a tag is encountered that occurs in the dictionary, its relevance is multiplied by the relevance in the tree and also added to all of the keywords in the tree above by multiplying with 0.5 each step higher up in the tree. This is done, because the keywords higher up in the tree can be triggered by hundreds of other keywords which leads to a very high relevance estimate for the keywords higher up. If all relevances are normalised at the end, the relevances of specific

keywords would be very low and only the top keywords would have a decent relevancy estimation. It also accounts for the fact that the keywords that were specifically triggered should be more relevant than indirectly triggered words. A book about football should get a high relevance for keywords “football” and “sport” and less relevance for “hobbies”.

The code for the recursive function is given below:

```

1 def recursive_func(tree):
2
3     if isinstance(tree, dict):
4         relevance_sum = 0
5         for tag, tree_sub in tree.items():
6             relevance = 0
7
8             if isinstance(tree_sub, float):
9                 if tag in tag_dict:
10                     relevance = tag_dict[tag] * tree_sub
11             else:
12                 # Check if context contains sub tags
13                 relevance = recursive_func(tree_sub)
14
15             if relevance > 0:
16                 tag_dict[tag] = relevance
17                 relevance_sum += relevance
18
19         return relevance_sum / 2
20
21 tag_dict = {}
22 for tag_obj in tag_list_custom:
23     tag_dict[tag_obj['tag']] = tag_obj['relevance']
24
25 recursive_func(config.TAG_MAP_INTERESTS)

```

Consider for example the following book description:

“He plays hockey with a hockey stick, but also plays golf sometimes.”

Looping over all terms in the tag\_word\_dict, there are two terms that get triggered:

- “hockey” is triggered twice, so the relevance of “hockey” is  $2 \cdot 1.4 = 2.8$
- “golf” is triggered once, so the relevance of “golf” is 1.3

The result is the following dictionary:

```

1 trigger_dict = {
2     "hockey": 2.8,
3     "golf": 1.3
4 }

```

After obtaining this dictionary, the tag\_map\_interests tree is walked. The first keyword is “hobbies” and its value is another dictionary. Since the value is another dictionary, the function is called again recursively, this time only on this smaller dictionary. Now, the first keyword encountered is “sport”, which

is a dictionary so the function is called recursively again. Next is the keyword “team sport” which also has a dictionary so another recursive call. The next keyword encountered is “voetbal”, which has a relevance instead of a dictionary. However, since this keyword is not in trigger\_dict, nothing happens. In the next iteration, “hockey” is encountered. This tag does occur in trigger\_dict, so its value (2.8) is multiplied with the value of “hockey” in the tag\_map\_interests tree (1.5) and the tag with this new relevance is added to a new dictionary:

```
1 tag_dict_final = {
2   "hockey": 4.2
3 }
```

The next keyword is “basketball”. Since this tag also does not occur in trigger\_dict, nothing is done. Since this was the last keyword in the “team sport” dictionary, the third recursive function call terminates, keeps half of the accumulated relevance of 4.2 (2.1) and adds the keyword to the dictionary:

```
1 tag_dict_final = {
2   "hockey": 4.2,
3   "team sport": 2.1
4 }
```

The next keyword is “solo-sport” which is a tree again so another recursive call is done. Only at the keyword “ golf ” the keyword is triggered and added to the dictionary with relevance  $1.3 \cdot 1.3$ :

```
1 tag_dict_final = {
2   "hockey": 4.2,
3   "team sport": 2.1,
4   " golf ": 1.69
5 }
```

When the keyword “ golf ” is reached, the third recursive call is terminated again. Half of 1.69 (0.845) is kept and the tag is added to the dictionary:

```
1 tag_dict_final = {
2   "hockey": 4.2,
3   "team sport": 2.1,
4   " golf ": 1.69,
5   "solo-sport": 0.845
6 }
```

Because “solo-sport” was the last key in the “sport” dictionary, the second recursive call is also terminated and half of the accumulated value (1.4725), 2.1 from team sport and 0.845 from solo-sport, is kept again and added to the dictionary:

```
1 tag_dict_final = {
2   "hockey": 4.2,
3   "team sport": 2.1,
4   " golf ": 1.69,
5   "solo-sport": 0.845,
6   "sport": 1.4725
7 }
```

Since none of the other keywords are triggered anymore, half of the accumulated value (1.4725/2) is returned and the keyword is added to the dictionary:

```

1 tag_dict_final = {
2     "hockey": 4.2,
3     "teamsport": 2.1,
4     " golf ": 1.69,
5     "solo-sport": 0.845,
6     "sport": 1.4725,
7     "hobbies": 0.73625
8 }

```

Finally, all values are normalised. Looking at the dictionary, 4.2 is the highest relevance so all values are divided by 4.2 which leads to the final result of:

```

1 tag_dict_final = {
2     "hockey": 1,
3     "teamsport": 0.5,
4     " golf ": 0.40238,
5     "solo-sport": 0.20119,
6     "sport": 0.350595,
7     "hobbies": 0.17530
8 }

```



## 2.5 AVI

The AVI score is a score that describes the difficulty of books. In Dutch, AVI stands for Analyse van Individualiseringsvormen, which translates to Analyses Of Individualization Forms. Most books in lower schools have the AVI score displayed on the cover, so children are able to choose books of their reading skill level. To obtain the AVI score of a book, the following table is utilized[9]:

Table 1: CILT

| AVI-niveau | CILT-waarde |
|------------|-------------|
| Start      | < 54.9      |
| M3         | 54.9 – 56.9 |
| E3         | 56.9 – 58.9 |
| M4         | 58.9 – 61.9 |
| E4         | 61.9 – 63.9 |
| M5         | 63.9 – 65.9 |
| E5         | 65.9 – 67.9 |
| M6         | 67.9 – 69.9 |
| E6         | 69.9 – 71.9 |
| M7         | 71.9 – 73.9 |
| E7         | 73.9 – 74.9 |
| Plus       | > 74.9      |

As can be seen, the AVI score depends on the CILT-waarde. In Dutch, CILT stands for Cito Index voor de LeesTechniek, which translates to Cito Index for the Reading Technique. This score is calculated using the following formula [9]:

$$\text{CILT-waarde} = 150 - (114.49 + 0.28 \cdot \text{frequentwords} - 12.33 \cdot \text{wordsize})$$

Here, the variable *frequentwords* is the percentage of words that occur in the list of 1000 most used words in the Dutch language and *wordsize* is the average length of the words.

Unfortunately, the precise list that Cito utilizes could not be found. To compensate for this, an online-sourced list [10] was utilized instead. The sole modification implemented was to remove all words that contain the character ‘\_’, and change the \$ character into a € sign.

## 2.6 CLIB-niveau

Cito also utilizes a measurement called CLIB-niveau, which determines how easy a text is to comprehend. CLIB stands for Cito LeesIndex Begrip, which is Dutch for Cito Reading Index Comprehension. To obtain the CLIB-niveau, the following table is utilized [9]:

Table 2: CLIB

| CLIB-niveau | CLIB-waarde                |
|-------------|----------------------------|
| Start       | $\leq -12$                 |
| 3           | $-12 < \text{CLIB} \leq 7$ |
| 4           | $7 < \text{CLIB} \leq 20$  |
| 5           | $20 < \text{CLIB} \leq 35$ |
| 6           | $35 < \text{CLIB} \leq 48$ |
| 7           | $48 < \text{CLIB} \leq 61$ |
| 8           | $61 < \text{CLIB} \leq 74$ |
| Plus        | $\text{CLIB} > 74$         |

The CLIB-waarde is calculated using the following formula [9]:

$$\begin{aligned}
 \text{CLIB-waarde} = & 46 - 6.603 \cdot \textit{wordsize} \\
 & + 0.474 \cdot \textit{frequentwords} \\
 & - 0.365 \cdot \textit{type/tokenratio} \\
 & + 1.425 \cdot \textit{sentencesize}
 \end{aligned}$$

Here, the variables *wordsize* and *frequentwords* are the same as in the CILT-waade calculation, *type/tokenratio* is the percentage of unique words with respect to the total amount of words in a text and *sentencesize* is the average amount of words in a sentence.

## 2.7 PDF-parser

As mentioned before, the scrapers are trying to determine the AVI score of a book. However, the AVI score is not always present, especially for books that are also targeting adults. Finding the CILT- and CLIB-waarde could help with making sure more books have this data available. However, to calculate these scores, the entire book is required in order to determine:

1. Percentage of frequent words
2. Average length of the words
3. Amount of unique words
4. Total amount of words
5. Average amount of words per sentence

Luckily, there is a solution. As can be observed in figure 2, the Titelbank file also contains PDF files. These PDF files usually contain the first few pages of the book, which contains a lot of useful data. To obtain useful data from these files, a PDF parser is utilized to extract the text from the PDF document.

Pdfminer [11] is a python library that can be used for parsing PDF files. The following code opens the PDF, iterates over its pages as well as the objects within the page [12]:

```
1 from pdfminer.pdfdocument import PDFDocument
2 from pdfminer.pdfpage import PDFPage
3 from pdfminer.pdfparser import PDFParser
4 from pdfminer.pdfinterp import PDFResourceManager,
5   PDFPageInterpreter
6 from pdfminer.converter import PDFPageAggregator
7 from pdfminer.layout import LAParams, LTTextBox, LTTextLine,
8   LTFigure
9
10 fp = open('example.pdf', 'rb')
11 parser = PDFParser(fp)
12 doc = PDFDocument(parser)
13
14 rsrcmgr = PDFResourceManager()
15 laparams = LAParams()
16 device = PDFPageAggregator(rsrcmgr, laparams=laparams)
17 interpreter = PDFPageInterpreter(rsrcmgr, device)
18 for page in PDFPage.create_pages(doc):
19     interpreter.process_page(page)
20     layout = device.get_result()
21     for obj in layout._objs:
22         # Do something with the object
```

To get a more comprehensive understanding of the code above, it is important to know the structure of objects that occur in a PDF, which is shown in the image below:

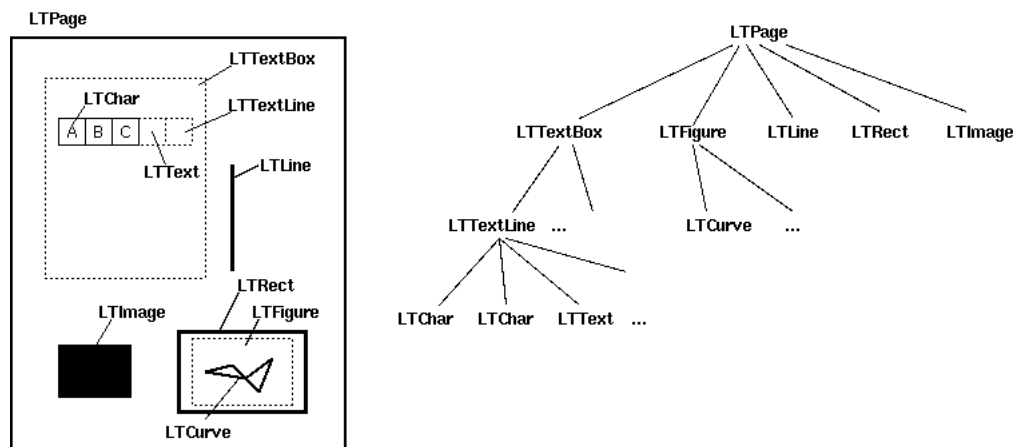


Figure 4: Layout of objects inside a pdf file  
Source <https://euske.github.io/pdfminer/layout.png>

Looking back at the code, it is observed that it iterates over all the pages in the PDF, and for each page another loop is executed for each object in the page. As shown in the image above, these objects can be one of the following: LTRect, LText, LImage, LFigure, and LTextLine.

Since there are children that enjoy reading books with lots of images, it might be useful to calculate the image per page ratio for the books in order to recommend books with lots of pictures to these children. The following code was implemented to obtain the desired result:

```
1 if isinstance(obj, LImage):
2     images_count += 1
```

To determine the unique and total amount of words, the words in the PDF need to be counted. As can be observed in figure 4, words occur in LText objects. Each LText object contains multiple LTextLine objects and each LTextLine object represents a line of words in the PDF. The code provided below accomplishes the intended purpose:

```
1 if isinstance(obj, LText):
2     for ob in obj:
3         if isinstance(ob, LTextLine):
4             words = ob.get_text().lower().strip().split(' ')
5             word_count += len(words)
```

To obtain the percentage of frequent words, another loop is utilized to loop over all words in the line. Then, for each word it checks if it occurs in the list of most frequent words and updates the counter accordingly as displayed below:

```
1 words = ob.get_text().lower().strip().split(' ')
2 for word in words:
3     if word in frequent_words:
4         frequent_word_count+=1
```

Although this would seem to work properly, in practise it does not. This is due to the fact that parsers are always tricky and need extra code for edge cases. In this case, the code does not account for the fact that there are a lot of punctuation marks in a sentence. If the word 'house' is one of the most frequently used words and there is a sentence in the PDF like: "Arjen walks towards the house.", the split function will extract "house." as word and checks if it occurs in the frequent word list, which it does not! To avoid this, all punctuation marks need to be eliminated which is accomplished by the code below:

```
1 to_be_replaced = {',': ' ', '-': ' ', ':': ' ', ';': ' ', ' ': ' ', '"': ' ',
2                  ' ': ' ', '?': ' ', '!': ' ', '\n': ' ', '_': ' '}
3 words = ob.get_text().lower().strip().split(' ')
4 for key, value in to_be_replaced.items():
5     words = [i.replace(key, value) for i in words]
```

Another variable that is extracted from the PDF is the font size. This is due to the fact that easier books tend to have a bigger font size than books for advanced readers. To obtain the font size, the code iterates over the objects in the LTTextLine object. These objects are of type LTChar of which both size as well as the name of the font are obtainable. All of this data is then stored in a dictionary, where the key is the concatenation of the font name and the font size and the value a dictionary that contains the amount of characters in this font, the amount of letters in this font as well as the amount of words in this font. This code is displayed below:

```
1 for character in ob:
2     if isinstance(character, LTChar):
3         font_size=round(character.size,2)
4         font_name=character.fontname
5         key = f'{font_size}_{font_name}'
6         if key in fonts_dict and character.get_text() != ' ' and
7         character.get_text() not in to_be_replaced:
8             fonts_dict[key]['letter_count']+=1
9             fonts_dict[key]['char_count']+=1
10        elif key in fonts_dict and (character.get_text() == ' ' or
11        character.get_text() in to_be_replaced):
12            fonts_dict[key]['char_count']+=1
13        elif key not in fonts_dict:
14            fonts_dict[key] = {}
15            fonts_dict[key]['letter_count'] = 1 if character.
16            get_text() != ' ' and character.get_text() not in
17            to_be_replaced else 0
18            fonts_dict[key]['char_count'] = 1
19            fonts_dict[key]['word_amount'] = 0
```

The sentence count is yet another variable that needs to be obtained in order to calculate the CILT and CLIB scores. This can easily be done by increasing the sentence count by one each time one of the following characters is encountered: ‘.’, ‘!’ or ‘?’. However, this does not account for the fact that there are lots of abbreviations used that contain the character ‘.’. Since a list of all abbreviations was not found, the code only considers abbreviations that occur in the list with the most used ones [13].

The final issue that need to be addressed is hyphenation. Consider the following sentence:

Djoerd did an amazing discovery and is excited to tell the news to his university department.

As can be seen, this sentence is covering two lines. This causes trouble for the following line in the code: `object.get_text().lower().strip().split(' ')`. When this line is executed and the punctuation marks are removed, the last word of the line will be “uni” and the first word of the next line will be “versity”. Not only will this lead to a miscount in the most frequent words, it will also mess up the total word count and hence also other variables that are dependent on the total word count.

A straightforward approach to handle this situation is to first concatenate all lines from all the pages. Then, in order to obtain the sentence count, the NLTK [14] package is utilized. This package contains the “`sent_tokenize()`” function which takes a string as input and returns a list of strings as output by applying various heuristics and rules to identify the boundaries between sentences. Since each string in the list now corresponds to a sentence, the sentence count can easily be calculated.

Another benefit of this approach is that the hyphenated word at the end of the line is now seen as one word with hyphen, meaning that one of the sentences after the tokenize function contains the string “uni-versity”. If the function: `“object.get_text().lower().strip().split(' ')”` is applied after the punctuation marks are removed per sentence, the result contains a list that contains only words without punctuation marks including hyphens and the word ‘university’ is correctly found:

[“Djoerd”, “did”, “an”, “amazing”, “discovery”, “and”, “is”, “excited”, “to”, “tell”, “the”, “news”, “to”, “his”, “university”, “department”]

The total amount of words as well as the percentage of frequent words are now easily obtainable as displayed in the code below:

```

1 import nltk
2
3 # Download the Dutch sentence model
4 nltk.download('punkt')
5 nltk.download('dutch')
6 pdf_text = ""
7
8 ..... # Some code before to iterate over all objects in the page
9     if isinstance(obj, LTTextBox):
10         for line in obj:
11             if isinstance(line, LTTextLine):
12                 pdf_text+=line.get_text()
13                 ... # Some other code for the font size
14
15 sentences = nltk.sent_tokenize(pdf_text, language='dutch')
16 sentence_count = len(sentences)
17 words = pdf_text.lower().strip().split(' ')
18 for word in words:
19     if word not in abbreviations:
20         for key, value in to_be_replaced.items():
21             word = word.replace(key, value)
22 word_count = len(words)
23 total_lenth = 0
24 unique_words = set()
25 for word in words:
26     if word in frequent_word_list:
27         frequent_word_count+=1
28     total_lenth+=len(word)
29     unique_words.add(word)
30 percentage_frequent_words = (frequent_word_count/word_count)*100
31 percentage_unique_words = (len(unique_words)/word_count)*100
32 average_word_length = total_lenth/word_count
33 words_per_sentence = word_count/sentence_count

```

All the necessary information to calculate AVI and CLIB-niveau, as well as information about images per page and font size are now extracted from the PDF and can be utilized by the recommendation system.

### 3 Recommendation Algorithm

There are many ways to recommend content to users. The most popular one is to recommend content based on popularity such as most minutes watched/listened, most liked/shared etc. Even though this is a strategy that can be applied universally, the recommendations are the same for everyone instead of being personalised for a specific user.

This is why companies like YouTube and Netflix utilize machine learning models in order to give their users the best personalised recommendations possible [15]. These machine learning models involve two major filtering components: content-based filtering and collaborative filtering which are discussed below.

#### 3.1 Content-based filtering

Content-based filtering, often called item-item collaborative filtering or classification based filtering, is a filter technique that makes decisions based on a correlation between products [15]. If a user finished reading a certain book, he will most likely want to read a book similar to the one he just read next. In Content-based filtering, similarities can be determined between the content the user consumed and all other content available. The algorithm then returns the content that is most similar to the one the user consumed. To clarify, an example is given below:

Table 3: Content-based filtering example

|          | Book A | Book B | Book C | Book D | Book E |
|----------|--------|--------|--------|--------|--------|
| Sci-fi   | ✓      |        |        | ✓      |        |
| Fantasy  | ✓      |        |        |        |        |
| Action   | ✓      |        | ✓      | ✓      |        |
| Horror   |        |        | ✓      |        | ✓      |
| Thriller |        | ✓      |        |        |        |
| Romance  |        | ✓      |        |        |        |
| Author A | ✓      |        | ✓      |        |        |
| Author B |        | ✓      |        |        |        |
| Author C |        |        |        | ✓      | ✓      |

As can be observed above, book A was written by author A and has genre sci-fi, fantasy and action. Suppose a user read this book and is looking for a good recommendation. It can be observed that books B and E are most-likely terrible recommendations considering the fact that none of the characteristics are the same. Books C and D on the other hand are more similar to book A. Book C because part of the genres overlap and it's written by the same author and book D because the genres are almost identical. So looking at the data, it is predicted that the user will like books C and D.



### 3.2 Collaborative filtering

Collaborative filtering, often called user-user collaborative filtering, is a filter technique based on the assumption that users with similar taste will probably like products that are consumed by the other [15]. If a user finished a book and talks to someone who also read that book, they can ask each other for a good recommendation because they have similar taste. While this similar taste is only based on reading the same book once, they might still prefer totally different things. However, when having multiple users with similar taste, this gets narrowed down:

Table 4: Collaborative filtering example

|        | Book A | Book B | Book C | Book D | Book E |
|--------|--------|--------|--------|--------|--------|
| User 1 | ✓      | ✓      | ✓      | ✗      | ✗      |
| User 2 | ✓      | ✗      | ✓      | ✗      | ✓      |
| User 3 | ✗      | ✗      | ✗      | ✓      | ✓      |
| User 4 | ✗      | ✗      | ✓      | ✓      | ✓      |
| User 5 | ✓      | ?      | ?      | ?      | ?      |

✓ read ✗ not read ? suggestions we want to determine

It is observed that users 1,2 and 5 all read book A. If these are considered users with similar taste, the following table can be generated:

|        | Book A | Book B | Book C | Book D | Book E |
|--------|--------|--------|--------|--------|--------|
| User 1 | ✓      | ✓      | ✓      | ✗      | ✗      |
| User 2 | ✓      | ✗      | ✓      | ✗      | ✓      |
| Count  |        | 1      | 2      | 0      | 1      |
| User 5 | ✓      | ?      | ?      | ?      | ?      |

✓ read ✗ not read ? probably a good suggestion

? probably a bad suggestion ? neither good nor bad

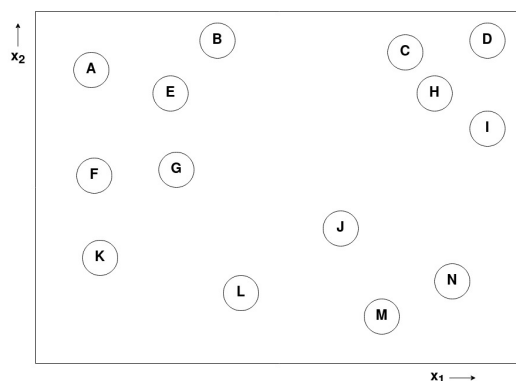
Looking at the table, it can be seen that book C is probably a good suggestion for user 5. It also shows that the more data is available, the better the recommendations will be. Suppose there was no information about user 2, then only user 1 was considered as similar and books B and C would have been recommended even though book B is generally read less often than book C and book E wouldn't even be considered even though it is read more often than the other books.

### 3.3 ANNOY

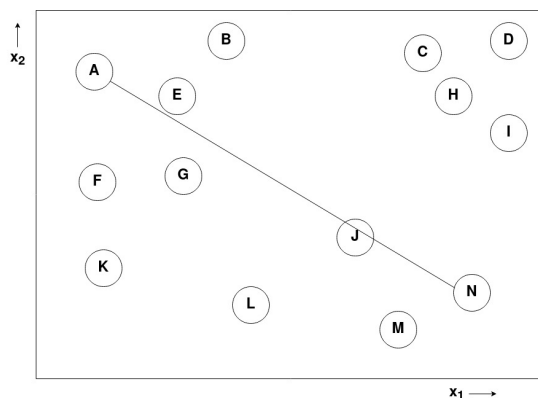
The aforementioned filtering methods both require the system to determine the best match. While finding the best match in the examples seemed easy, finding them in a larger data set with more dimensions and more data points is a lot more difficult. In order to solve this, ANNOY can be used.

ANNOY, Approximate Nearest Neighbors Oh Yeah, is an algorithm based on random projections and trees. It was developed by Erik Bernhardsson in 2015 who worked at Spotify at the time [16]. What makes this algorithm useful, is the fact that it does not try to calculate the exact nearest neighbours. It rather, as the name suggests, finds the approximately closest neighbours which results in a huge complexity reduction.

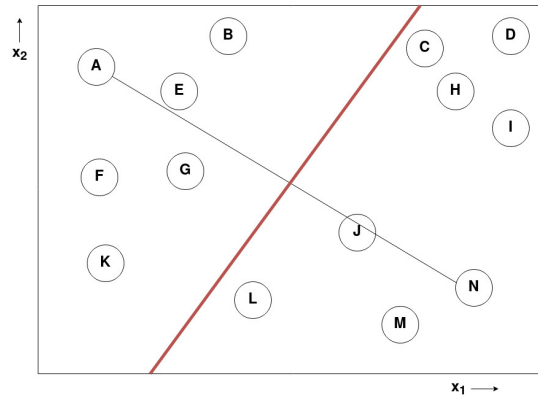
This complexity reduction comes from the fact that a tree is constructed from the data in which searching for the nearest neighbour can be done in logarithmic time. This is explained in more detail below:



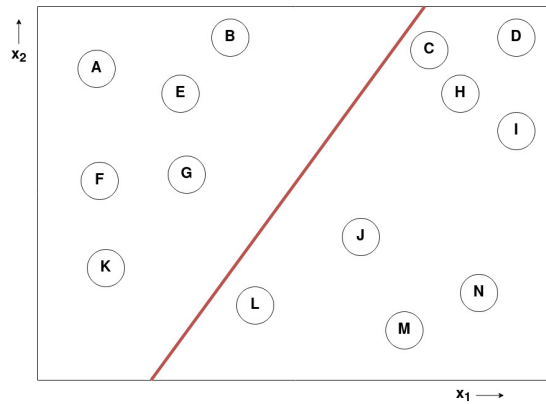
As can be seen, there are 14 data points in two dimensional space. The first thing that is done is to take two random points, in this case A and N, and connect them with a line:



Next, a new line is drawn perpendicular to the one from A to N that crosses in the middle:

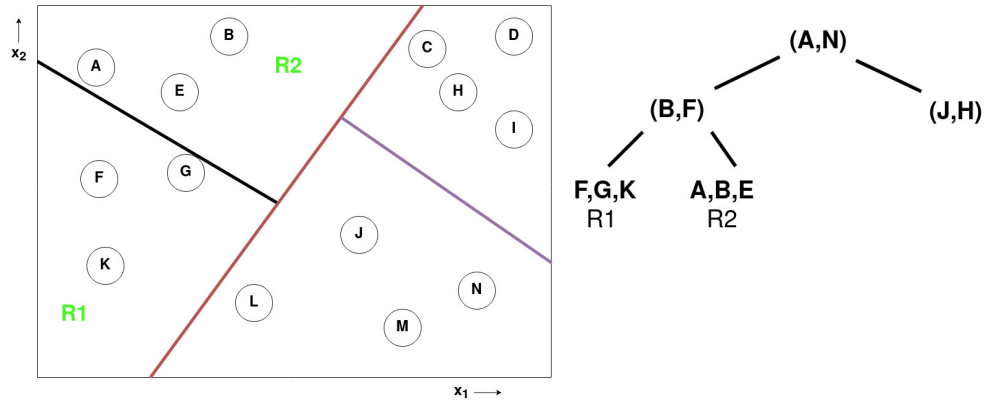


The line between points A and N is removed which results in two different regions. At the same time, a tree structure is generated where (A,N) means that the data was split between points A and N:

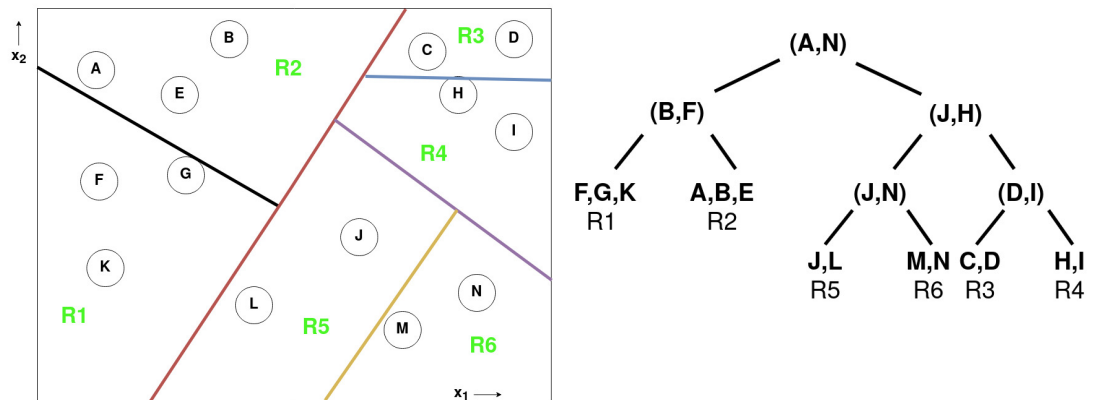


**(A,N)**

The algorithm continues splitting each area as done above and terminates when the area contains less than a predetermined number of points, 3 in this example.



Regions R1 and R2 contain 3 or less data points so splitting in these areas is terminated.



All regions now contain 3 or less data points and dividing data points into regions is terminated. In order to determine the nearest neighbour of a point, the algorithm looks at the region that the point is in and only calculates the distance between the points in this region. So if one intends to determine the closest neighbour of M, the following is done:

1. Is M left or right of the line between points A and N?  
Right, so proceed to the right in the tree.
2. Is M left or right of the line between points J and H?  
Left, so proceed to the left in the tree.
3. Is M left or right of the line between points J and N?  
Right, so proceed to the right in the tree.
4. End of the tree is reached since there are no more splits in this branch. The closest neighbour is now determined by calculating the distances between M and all other points in this region. Since N is the only other point in this region, it is also the nearest neighbour of M.

Because the regions are randomly generated, it could happen that the tree that was generated does not represent every region equally well. Looking back at the example above, it can be seen that C is closest to H. It can also be seen that this point is not considered because it is in a different region and hence the algorithm returns I as nearest neighbour. It is desired to minimise these errors as much as possible, which is solved by generating a forest of multiple trees instead of just a single tree. The forest contains multiple trees, each with its own splits. After the forest is generated, the nearest neighbours from all trees are compared and the best one is returned. In most ANNOY implementations today, additional heuristics are used to improve the algorithm even further.

### 3.4 Different metric options

In the example above, the euclidean distance was utilized as a measure to determine the nearest neighbours of the data points. However, other metric options that ANNOY supports can be considered in order to acquire the best results. These different metric options are listed below [17][18]:

1. Euclidean distance, which is the straight-line distance between 2 points. It is calculated as follows:  $\sqrt{\sum_{i=1}^n (p_i - q_i)^2}$
2. Manhattan distance, which is the distance between two points measured along the axes at right angles. It is calculated as follows:  $\sum_{i=1}^n |p_i - q_i|$

Visual representation:

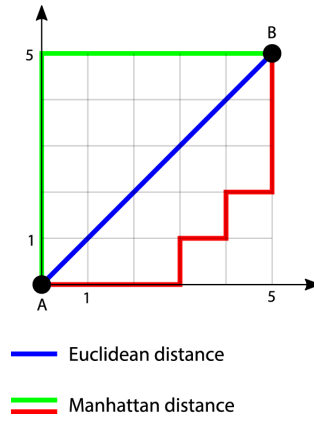


Figure 5: Euclidean and Manhattan distances

Link <https://www.researchgate.net/publication/333430988>

3. Angular distance, which measures the angle between two vectors. It is calculated as follows:  $\arccos\left(\frac{\langle p, q \rangle}{\|p\| \cdot \|q\|}\right)$
4. Dot product distance, which measures the cosine of the angle between two vectors. It is calculated as follows:  $1 - \frac{\langle p, q \rangle}{\|p\| \cdot \|q\|}$

Visual representation:

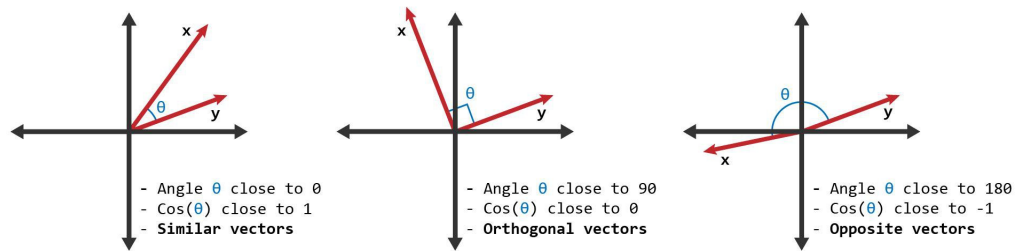


Figure 6: Cosine-similarity

Link: <https://storage.googleapis.com/lds-media/images/cosine-similarity-vectors.original.jpg>

As can be observed above, the more similar the vectors are, the smaller the angle and the more the cosine of the angle approaches 1. Because the distance should be the smallest for similar vectors, 1 minus the result is utilized instead in order to obtain a good distance representation for the dot product distance. With angular distance this is not necessary because the distance measure is the angle itself, which is already 0 for similar vectors and becomes larger the more the vectors differ.

5. Hamming distance, which is the number of substitutions needed to turn the first item into the second item.

It is calculated as follows:  $\sum_{i=1}^n [p_i \neq q_i]$

Visual representation:

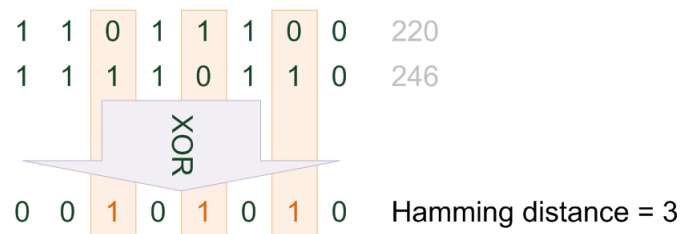


Figure 7: Hamming distance

Link: [www.impuls-imaging.com/wp-content/uploads/2013/07/Hamming.png](http://www.impuls-imaging.com/wp-content/uploads/2013/07/Hamming.png)

As can be seen above, the number of substitutions can also be obtained by applying the XOR operation if the representation of the items is a binary number.

### 3.5 Curse of dimensionality

Having now witnessed the different metric options that ANNOY supports, the best one needs to be chosen. In order to do so, the following example is utilized with vectors  $v1 = [1, 4]$ ,  $v2 = [8, 8]$  and  $v3 = [3, 1]$ :

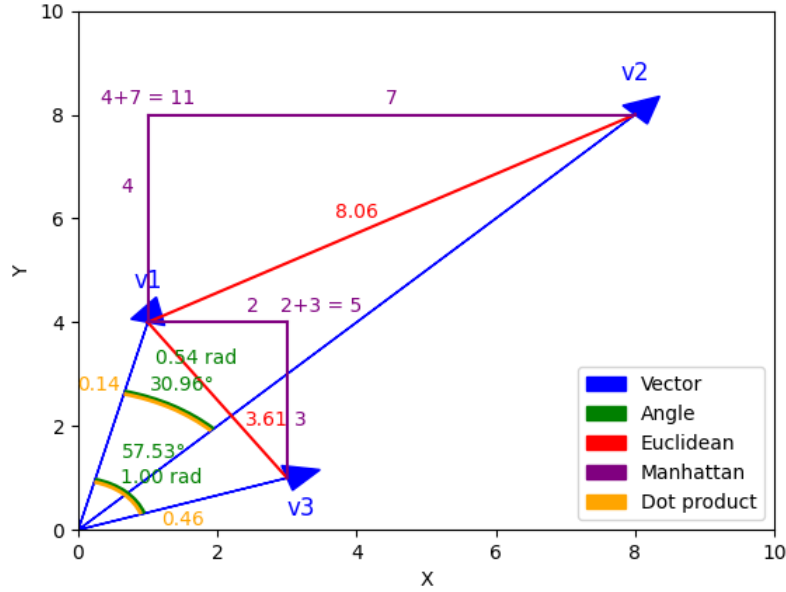


Figure 8: Angular vs Euclidean vs Manhattan vs Dot product distance  
See Appendix C for the code used to generate this

The Hamming distance is not shown, but it is 2 for all vectors because all vectors have different x and y coordinates. It is observed that the euclidean distance between  $v1$  and  $v3$  is smaller than the euclidean distance between  $v1$  and  $v2$ . This is also the case with the Manhattan distance. However, looking at the angle between  $v1$  and  $v3$ , it can be observed that it is larger than the angle between  $v1$  and  $v2$ . The dot product distance shows this even better, since the dot product distance between  $v1$  and  $v3$  is almost triple the size of  $v1$  and  $v2$ . So is  $v1$  considered most similar to  $v2$  or  $v3$ ?

Although all discussed options are good measurements, there is an important factor to take into account. As already mentioned before, there are cases where certain distance metrics are not a good measure for distance. These cases occur in high dimensional space. In high dimensional space, data becomes more sparse[19]. This increased sparseness leads to an increase in the minimum distance between two points in the data set. In other words, for every dimension added to the data, the minimum distance increases up to a point where there



is almost no difference between the minimum and maximum distance anymore [20]. This phenomenon is known as the curse of dimensionality.

To see why this happens, let us start simple with a few data points in 1 dimensional space:

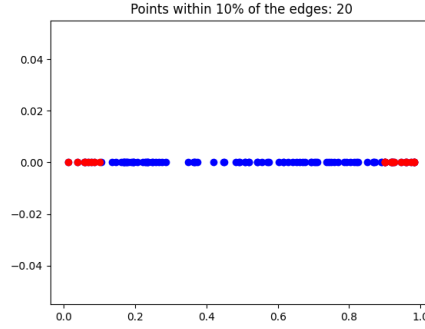


Figure 9: Random points in 1D space  
See Appendix B.1 for the code used to generate this

The image above was created by generating 100 random points in 1 dimensional space. All points near the edges, points within 10% of the edge, are coloured red. As we would expect, 20 points fall within 10% of the edges, which is equivalent to 20% of the points.

The same is done for 2 dimensional space:

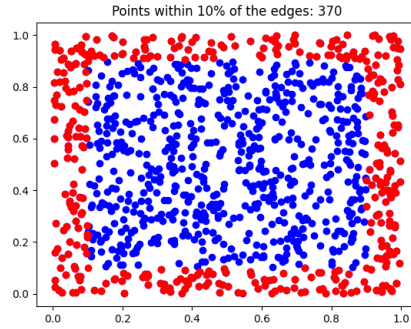


Figure 10: Random points in 2D space  
See Appendix B.2 for the code used to generate this

In order to keep the sparseness the same, 1000 random points are now generated instead of 100. As can be seen, there are now 370 points within 10% of the edges, which is almost twice as many!

Looking at 3 dimensional space, the number is still increasing:

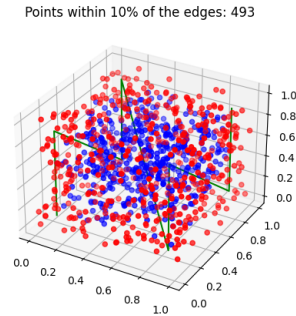


Figure 11: Random points in 3D space  
See Appendix [B.3](#) for the code used to generate this

This image was generated using 1000 random points, because with the same sparseness as in the previous 2 images, the image would have been an almost solid red block and no blue points would be visible because they would all be behind the red points. As can be seen, there are now 493 points near the edges. That's almost 50% of the total amount of data points!

Upon further examination of data with more dimensions, the number of points near the edges keeps increasing [\[19\]](#):

| Number of Dimensions | Percentage of data within 10% of the edges |
|----------------------|--|
| 4                    | 60   |
| 5                    | 68   |
| 6                    | 74   |
| 7                    | 80   |
| 8                    | 82   |
| 9                    | 87   |

So what is the underlying reason behind this phenomenon?

### 3.6 Limit/inside ratio

Looking at this from a different perspective, a mathematical pattern appears that clarifies this behaviour. Instead of looking at the points near the edges, only the points exactly at the edge are considered:

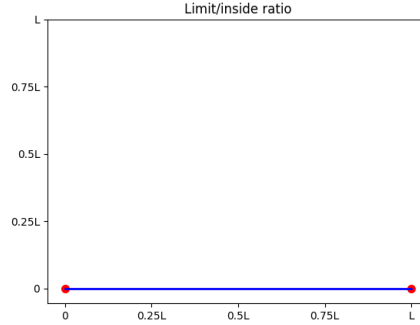


Figure 12: Limit/inside ratio in 1D space  
See Appendix D.1 for the code used to generate this

As can be observed there are only 2 points at the limit (point at 0 and point at L) and all other points are on the inside. All points in between are inside these limits, hence the name limit/inside ratio. In general, for a 1 dimensional space of length L, there are 2 points at the limit: 0 and L and all other points lay somewhere in between where they have L different options. In other words, the limit/inside ratio for 1 dimension =  $2/L$ .

For 2 dimensions something similar is observed:

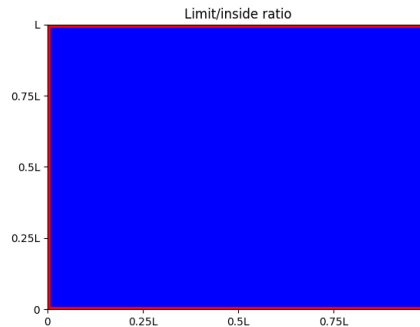


Figure 13: Limit/inside ratio in 2D space  
See Appendix D.2 for the code used to generate this

The limit is now  $4L$ , because there are 4 sides of length  $L$ , and the inside is  $L^2$ . This results in a limit/inside ratio of  $\frac{4L}{L^2} = 4/L$  [19].

To ensure the continuity of this pattern, an analysis of the third dimension is needed:

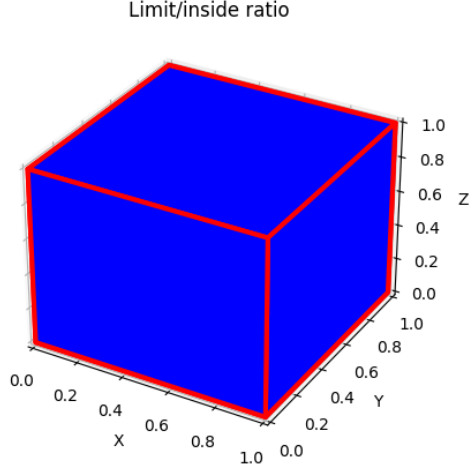


Figure 14: Limit/inside ratio in 3D space  
See Appendix D.3 for the code used to generate this

The cube has 6 faces as the limit, each with a size of  $L$  by  $L$  meaning the limit is  $6L^2$ . The inside of the cube is  $L^3$  which results in a limit/inside ratio of  $\frac{6L^2}{L^3} = 6/L$  [19].

The mathematical pattern is now visible, for every dimension added the limit/inside ratio increases in accordance with the following formula:

$$ratio = \frac{2D}{L}$$

Where  $D$  is the amount of dimensions and  $L$  the size of the dimensions.

Looking at the formula it follows that the data is more likely to be near the limits of the space in higher dimensions.

### 3.7 Calculations

Having gained insight on the curse of dimensionality, it becomes evident that not every distance measurement is suited for estimating similarities in higher dimensional space. However, since a measurement to see which points are closest to each other is still required, a measurement option that is still affected by a small changes in high dimensional vectors is required. Using the different options as discussed in section 3.4, distance calculations were made and compared with the following two vectors in 512 dimensional space:

$$v1 = [0, 1, 0, 1, 0, 1, 0, 1, \dots, 0, 1]$$

$$v2 = [0, 0, 1, 1, 0, 0, 1, 1, \dots, 1, 1]$$

$v1$  is a vector that alternates between zero and one,  $v2$  is a vector that alternates between two zeros and two ones. Calculations for the distance between these vectors using different methods are provided below, as well as the new distance if one vector changes one 0 to a 1 and the percentage change.

#### 3.7.1 Euclidean

$$\sqrt{\sum_{i=1}^{512} (p_i - q_i)^2} = \sqrt{0 + 1 + 1 + 0 + 0 + 1 + 1 + 0 + \dots + 0 + 1 + 1 + 0} = \sqrt{256} = 16$$

If  $v1$  now changes the first 0 into a 1, the distance would be  $\sqrt{257}$ , which is an increase of  $\frac{\sqrt{257}-16}{16} \cdot 100\% \approx 0.2\%$

#### 3.7.2 Manhattan distance

$$\sum_{i=1}^n |p_i - q_i| = 0 + 1 + 1 + 0 + 0 + 1 + 1 + 0 + \dots + 0 + 1 + 1 + 0 = 256$$

If  $v1$  now changes the first 0 into a 1, the distance would be 257, which is an increase of  $\frac{257-256}{256} \cdot 100\% \approx 0.4\%$

#### 3.7.3 Angular distance

$$\begin{aligned} \arccos\left(\frac{\langle p, q \rangle}{\|p\| \cdot \|q\|}\right) &= \arccos\left(\frac{0 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 1 + \dots + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 1}{\sqrt{0^2 + 1^2 + 0^2 + 1^2 + \dots + 0^2 + 1^2} \cdot \sqrt{0^2 + 0^2 + 1^2 + 1^2 + \dots + 1^2 + 1^2}}\right) \\ &= \arccos\left(\frac{128}{\sqrt{256} \cdot \sqrt{256}}\right) \\ &= \arccos\left(\frac{128}{256}\right) \\ &= \arccos\left(\frac{1}{2}\right) \\ &= 60^\circ \end{aligned}$$

If  $v1$  now changes the first 0 into a 1, the angle would be  $\arccos\left(\frac{128}{\sqrt{257} \cdot \sqrt{256}}\right) = 60.6^\circ$ , which is an increase of  $\frac{60.6-60}{60} \cdot 100\% = 1\%$

### 3.7.4 Dot product distance

$$\begin{aligned}
1 - \frac{\langle p, q \rangle}{\|p\| \cdot \|q\|} &= 1 - \frac{0 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 1 + \dots + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 1}{\sqrt{0^2 + 1^2 + 0^2 + 1^2 + \dots + 0^2 + 1^2} \cdot \sqrt{0^2 + 0^2 + 1^2 + 1^2 + \dots + 1^2 + 1^2}} \\
&= 1 - \frac{128}{\sqrt{256} \cdot \sqrt{256}} \\
&= 1 - \frac{128}{256} \\
&= 1 - \frac{1}{2} \\
&= \frac{1}{2}
\end{aligned}$$

If  $v_1$  now changes the first 0 into a 1, the angle would be  $1 - \frac{128}{\sqrt{257} \cdot \sqrt{256}} \approx 0.50097$ , which is an increase of  $\frac{0.50097 - 0.5}{0.5} \cdot 100\% \approx 0.2\%$

### 3.7.5 Hamming distance

$$\sum_{i=1}^n [p_i \neq q_i] = 0 + 1 + 1 + 0 + 0 + 1 + 1 + 0 + \dots + 0 + 1 + 1 + 0 = 256$$

If  $v_1$  now changes the first 0 into a 1, the distance would be 257, which is an increase of  $\frac{257 - 256}{256} \cdot 100\% \approx 0.4\%$

### 3.7.6 Conclusion

From the calculations above, it becomes evident that the curse of dimensionality is less impactful for the angular distance between vectors and more impactful for the euclidean distance between vectors [21]. Since a metric option that is least affected by the curse of dimensionality is required, angular distance is the best option which is also the one used in the final implementation of the recommendation system.

## 4 Implementation

In the next section the acquired data about books is covered followed by a section that describes how this data is preprocessed in order for it to be used in the recommendation system. The last section will cover the implementation of the recommendation system itself.

### 4.1 Database structure

The result of utilizing the aforementioned methods to gather data about books from different sources is the following database model:

| Variable Name             | Variable type     |
|---------------------------|-------------------|
| ISBN13                    | CHAR(13)          |
| Page count                | SMALLINT unsigned |
| Book title                | VARCHAR(256)      |
| Book type                 | TINYINT unsigned  |
| Book language             | TINYINT unsigned  |
| Dyslexia                  | BOOLEAN           |
| Minimum age               | TINYINT unsigned  |
| Maximum age               | TINYINT unsigned  |
| AVI                       | SMALLINT unsigned |
| Words per page            | DECIMAL(8, 4)     |
| Words per sentence        | DECIMAL(8, 4)     |
| Images per page           | DECIMAL(8, 4)     |
| Primary font size         | DECIMAL(8, 4)     |
| Average font size         | DECIMAL(8, 4)     |
| Percentage frequent words | DECIMAL(8, 4)     |
| Type/token ratio          | DECIMAL(8, 4)     |
| Average word length       | DECIMAL(8, 4)     |
| Release date              | DATE              |
| Keywords                  | ARRAY             |

Table 5: Variables about books in database

For book type, book language and AVI, python enumerations[22] are used:

```
1 from enum import Enum
2
3 class BookType(Enum):
4     LEESBOEK = 0
5     INFORMATIEF = 1
6     PRENTENBOEK = 2
7     STRIPBOEK = 3
8     DICHTBUNDEL = 4
9     VOORLEESBOEK = 5
10    ORIENTATIE_OP_LEZEN = 6
11    ZOEKBOEK = 7
12    AANWIJSBOEK = 8
13
14 class BookLanguage(Enum):
15     DUTCH = 0
16     ENGLISH = 1
17     GERMAN = 2
18     FRENCH = 3
19     SPANISH = 4
20     FRYSIAN = 5
21
22 class Avi(Enum):
23     START = 0
24     M3 = 1
25     E3 = 2
26     M4 = 3
27     E4 = 4
28     M5 = 5
29     E5 = 6
30     M6 = 7
31     E6 = 8
32     M7 = 9
33     E7 = 10
34     PLUS = 11
```

## 4.2 Data preparation

The ANNOY algorithm finds the approximately closest neighbours of data points. The idea is that all books in the database are mapped to a data point in some high dimensional space and letting ANNOY figure out the closest neighbours. In order to accomplish this, a dimension is added for every variable in the database, except for ISBN-13 and book title:

isbn13 = [Page count, Book type, Book language, Dyslexia, Minimum age, Maximum age, AVI, Words per page, Words per sentence, Images per page, Primary font size, Average font size, Percentage frequent words, Type/token ratio, Average word length, Release date, keywords]



For example, in the given situation where there is a German book with ISBN-13 9781234567890 of 400 pages, minimum age of 12 with an average of 250 words per page, the following vector is generated:

$$9781234567890 = [400, 0, 2, 0, 12, 0, 0, 250, 0, 0, 0, 0, 0, 0, 0, 0]$$

However, there exists a huge difference between the variables page count and minimum age. This can be explained by the fact that these variables operate on a different scale, so they are difficult to compare with one another. However, both of these variables are important data for books so they have to be considered when recommending books. To solve this, the data is scaled in such a way that all values are between 0 and 1 for all variables. The AVI level will always be an integer between 0 and 11, so scaling that will be easy. Unfortunately, this doesn't work for page count. If the max page count is set to the book with the most pages, there might be a book someday with more pages and the maximum should be adjusted again. Another problem is the array of keywords. An array can't be compared to integers.

In order to deal with all of this, every variable is split into multiple dimensions with 1 as maximum value and 0 as minimum value which can be seen in appendix A. For age, the following code was utilized:

```

1 age_vector = 16 * [0]
2 if item['min_age'] is not None:
3     min_age = cap(15, item['min_age'])
4     if item['max_age'] is None:
5         age_vector[cap(15, min_age - 2)] = 0.25
6         age_vector[cap(15, min_age - 1)] = 0.5
7         age_vector[cap(15, min_age + 1)] = 0.5
8         age_vector[cap(15, min_age + 2)] = 0.5
9         age_vector[cap(15, min_age)] = 1
10    else:
11        max_age = cap(15, item['max_age'])
12        age_vector[cap(15, min_age - 2)] = 0.25
13        age_vector[cap(15, min_age - 1)] = 0.5
14        age_vector[cap(15, max_age + 1)] = 0.5
15        age_vector[cap(15, max_age + 2)] = 0.5
16        age_vector[min_age:max_age] = [1] * (max_age - min_age)

```

Since the keywords are already a list of values, the list can just be extended by the list with the values for all the tags. The result is a dictionary with the isbn13 numbers as keys and as value a data point in approximately 450 dimensional space, which can be used in ANNOY.

### 4.3 Recommendation formula

While there is nothing wrong with supplying 450 dimensional points to ANNOY, it is most likely a problem that of the 450 available dimensions, 400 are used by the tags meaning books with a few similar tags will already match better with each other independent of the age variable meaning someone that reads Dolfje Weerwolfje could get the Twilight Saga recommended.

To guarantee that the keywords do not dominate the recommendations, the recommendation is split into multiple categories using the following formula:

$$\alpha_1 \cdot x_1 + \alpha_2 \cdot x_2 + \alpha_3 \cdot x_3 + \alpha_4 \cdot x_4$$

Where  $\alpha_i$  is the weight attached to part  $i$  of the formula,  $x_1$  is the ANNOY result of the tags,  $x_2$  is the ANNOY result of the age,  $x_3$  is the ANNOY result of the avi score and  $x_4$  is the ANNOY result of remaining variables like page count and pdf parser results like words per page, image count etc.

Having separated the different categories in this way, each category is assigned its own weight to determine how important each category is for the end result. To ensure the end result is between 0 and 1, it's important that  $\sum_{i=1}^4 \alpha_i = 1$ . The only thing that needs to change to achieve this is split the generated array in 4 parts, one per category.

The Annoy library is used as follows:

```
1 from annoy import AnnoyIndex
2
3 class GeneratorAnnoy:
4
5     def __init__(self, vector_dict: dict):
6         self.vector_dict = vector_dict
7         self.vector_list = list(self.vector_dict.values())
8         self.vector_length = len(self.vector_list[0])
9
10        # Generate conversion tables
11        self.obj_id_to_index = {}
12        self.index_to_obj_id = {}
13        for index, obj_id in enumerate(list(self.vector_dict.keys())):
14            self.obj_id_to_index[obj_id] = index
15            self.index_to_obj_id[index] = obj_id
16
17        def build_forest(self):
18            # Init list of item vectors
19            self.annoy_index = AnnoyIndex(self.vector_length, 'angular')
20            for i, vector in enumerate(self.vector_list):
21                self.annoy_index.add_item(i, vector)
22
23            # Build forest of TREE_COUNT trees
24            self.annoy_index.build(config.TREE_COUNT)
25
26        def save_annoy_index(self):
27            """Store forest in file"""
28
```

```

29     self.annoy_index.save('test.ann')
30
31     def compute_distances(self, obj_id: int, count: int):
32         """
33         Compute nearest neighbours given an obj_id.
34
35         :param int obj_id: the id of the object to which the
36         distances should be computed.
37
38         :return list indices: the indices of the nearest neighbours
39         .
40         :return list distances: the distances to the nearest
41         neighbours.
42         """
43
44         index = self.obj_id_to_index[obj_id]
45         result = self.
46
47     def compute_similarities(self, count: int):
48         similarities = {}
49         for obj_id in self.vector_dict.keys():
50             indices, distances = self.compute_distances(obj_id,
51             count)
52
53             # Create similarities dict
54             # Template: dict[base_id][recommendation_id] = distance
55             similarities[obj_id] = {}
56             for i in range(count):
57                 index = indices[i]
58                 similarities[obj_id][self.index_to_obj_id[index]] =
59                 distances[i]
60
61         return similarities

```

The GeneratorAnnoy class is then used as follows to calculate the closest neighbours based on age:

```

1 def compute_similarities(vector_dict: dict, count: int):
2     count = count if count < len(vector_dict) else len(vector_dict)
3
4     generator = GeneratorAnnoy(vector_dict)
5     generator.build_forest()
6     similarities = generator.compute_similarities(count)
7
8     return similarities
9
10 age_similarities = compute_similarities(
11     edition_to_age_vector_dict,
12     config.CALCULATION_RECOMMENDATION_COUNT
13 )

```

The result is a dictionary with as keys the ISBN-13 number of the book and as value a list of ISBN-13 numbers of recommendations in order. The same thing is done for the other categories to obtain the closest neighbours for every ISBN-13 per category. In order to apply the aforementioned formula, it is important to keep in mind that all of the categories can have different ISBN-13 recommendations. In order to deal with this correctly, the following code is

utilized:

```
1 recommendation_scores = {}
2 for id in keyword_recommendations.keys():
3     recommendation_scores[id] = {'score': 0, 'tags': 0, 'age': 0, '
    avi': 0, 'remainder': 0}
4 for id, score in keyword_recommendations.items():
5     recommendation_scores[id]['tags'] = 1-(score/2)
6 for id in age_recommendations.keys():
7     if id not in recommendation_scores:
8         recommendation_scores[id] = {'score': 0, 'tags': 0, 'age':
    0, 'avi': 0, 'remainder': 0}
9 for id, score in age_recommendations.items():
10    recommendation_scores[id]['age'] = 1-(score/2)
11 for id in avi_recommendations.keys():
12    if id not in recommendation_scores:
13        recommendation_scores[id] = {'score': 0, 'tags': 0, 'age':
    0, 'avi': 0, 'remainder': 0}
14 for id, score in avi_recommendations.items():
15    recommendation_scores[id]['avi'] = 1-(score/2)
16 for id in remainder_recommendations.keys():
17    if id not in recommendation_scores:
18        recommendation_scores[id] = {'score': 0, 'tags': 0, 'age':
    0, 'avi': 0, 'remainder': 0}
19 for id, score in remainder_recommendations.items():
20    recommendation_scores[id]['remainder'] = 1-(score/2)
21
22 for id in recommendation_scores.keys():
23    recommendation_scores[id]['score'] = \
24        0.8*recommendation_scores[id]['tags']+\\
25        0.1*recommendation_scores[id]['age']+\\
26        0.05*recommendation_scores[id]['avi']+\\
27        0.05*recommendation_scores[id]['remainder']
```

The code above iterates over all recommendations per category and inserts the score in the keyword\_recommendations dictionary under the specific category. Because the Annoy library returns a score between 0 and 2, the score needs to be divided by 2 to get a result between 0 and 1. Also, because more similar vectors have a lower score,  $1-(\text{score}/2)$  is applied in order to obtain the desired result, namely similar vectors with no angle between them have a score of 1. To obtain the final recommendation score, the score per category is taken and multiplied by their respective weights. These weights were the result of much testing and can still change over time. To give an example, when the weight for the age category was too high, there were some books that were recommended for almost every other book, which was most likely the case because they had a large age range that made them match with most books.

## 5 Results

After implementing the recommendation system in the app, the results were promising:



Figure 15: Dolfje Weerwolfje recommendations in the app

As can be seen, the book about Dolfje Weerwolfje gets recommendations about other books with werewolves, most of these are also Dolfje Weerwolfje. Looking at the database, the scores that a recommendation gets in every category can be obtained:

| isbn13_book   | overall_score | tags_score | age_score | avi_score | remainder_score | sequence_number | isbn13        |
|---------------|---------------|------------|-----------|-----------|-----------------|-----------------|---------------|
| 9789025861346 | 0.8221        | 0.7983     | 1.0000    | 0.6445    | 0.5685          | 0               | 9789025851200 |
| 9789025861346 | 0.8221        | 0.7983     | 1.0000    | 0.6445    | 0.5685          | 0               | 9789025867614 |
| 9789025861346 | 0.8182        | 0.7933     | 1.0000    | 0.6445    | 0.5685          | 1               | 9789025856663 |
| 9789025861346 | 0.7920        | 0.7402     | 1.0000    | 0.6445    | 0.8672          | 2               | 9789025862800 |
| 9789025861346 | 0.7920        | 0.7402     | 1.0000    | 0.6445    | 0.8672          | 2               | 9789025875626 |
| 9789025861346 | 0.7896        | 0.7402     | 1.0000    | 0.6445    | 0.8187          | 3               | 9789025861759 |
| 9789025861346 | 0.7896        | 0.7402     | 1.0000    | 0.6445    | 0.8187          | 3               | 9789025866402 |
| 9789025861346 | 0.7896        | 0.7402     | 1.0000    | 0.6445    | 0.8187          | 3               | 9789025881108 |
| 9789025861346 | 0.7860        | 0.7418     | 1.0000    | 1.0000    | 0.7224          | 4               | 9789025869380 |
| 9789025861346 | 0.7860        | 0.7418     | 1.0000    | 1.0000    | 0.7224          | 4               | 9789025877842 |
| 9789025861346 | 0.7810        | 0.7180     | 1.0000    | 0.6445    | 0.9896          | 5               | 9789025880897 |
| 9789025861346 | 0.7702        | 0.7287     | 1.0000    | 0.3334    | 0.6082          | 6               | 9789048736645 |
| 9789025861346 | 0.7665        | 0.7314     | 1.0000    | 0.6445    | 0.4941          | 7               | 9789025860097 |
| 9789025861346 | 0.7625        | 0.7213     | 1.0000    | 0.2929    | 0.5685          | 8               | 9789025845858 |
| 9789025861346 | 0.7625        | 0.7213     | 1.0000    | 0.2929    | 0.5685          | 8               | 9789025866648 |
| 9789025861346 | 0.7595        | 0.6983     | 1.0000    | 0.3334    | 0.8663          | 9               | 9789025870584 |
| 9789025861346 | 0.7590        | 0.7169     | 1.0000    | 0.2929    | 0.5685          | 10              | 9789025855277 |
| 9789025861346 | 0.7590        | 0.7169     | 1.0000    | 0.2929    | 0.5685          | 10              | 9789025880170 |
| 9789025861346 | 0.7581        | 0.7058     | 1.0000    | 1.0000    | 0.7223          | 11              | 9789025881122 |
| 9789025861346 | 0.7292        | 0.6509     | 1.0000    | 0.6445    | 0.9953          | 12              | 9789025877316 |
| 9789025861346 | 0.7253        | 0.6636     | 1.0000    | 0.6445    | 0.7208          | 13              | 9789025849245 |
| 9789025861346 | 0.7253        | 0.6636     | 1.0000    | 0.6445    | 0.7208          | 13              | 9789025863029 |
| 9789025861346 | 0.7253        | 0.6636     | 1.0000    | 0.6445    | 0.7208          | 13              | 9789025876159 |
| 9789025861346 | 0.7231        | 0.6754     | 1.0000    | 1.0000    | 0.4941          | 14              | 9789045120188 |
| 9789025861346 | 0.7201        | 0.6505     | 1.0000    | 0.6445    | 0.8185          | 15              | 9789025876814 |
| 9789025861346 | 0.7166        | 0.7190     | 0.6772    | 0.6445    | 0.8180          | 16              | 9789025860684 |
| 9789025861346 | 0.7151        | 0.6603     | 1.0000    | 0.6445    | 0.5685          | 17              | 9789048807154 |
| 9789025861346 | 0.7137        | 0.6423     | 1.0000    | 1.0000    | 0.8192          | 18              | 9789025873035 |
| 9789025861346 | 0.7137        | 0.6423     | 1.0000    | 1.0000    | 0.8192          | 18              | 9789025883133 |
| 9789025861346 | 0.7096        | 0.6432     | 1.0000    | 1.0000    | 0.7220          | 19              | 9789025842185 |
| 9789025861346 | 0.7096        | 0.6432     | 1.0000    | 1.0000    | 0.7220          | 19              | 9789025862787 |
| 9789025861346 | 0.7014        | 0.6425     | 1.0000    | 0.6445    | 0.5685          | 20              | 9789048802173 |

Figure 16: Scores saved in database for Dolfje Weerwolfje

As can be seen, all of the recommended books match 100 percent on age. This can be explained by the fact that age has a large weight within the recommendation formula. After all, Twilight should not be recommended to children of 8 years old. It can also be seen that there are sometimes multiple ISBN-13 recommendations per sequence number. This happens because some books get multiple editions meaning the books are almost identical except for a few changes. In this case, the ISBN-13 numbers will be mapped to the same book in the recommendation system. This also prevents that a different edition of the same book gets recommended. At the moment of writing, AVI does not have a large weight in the recommendation formula, because this data is unknown or imprecise for most books. In section 6 a solution is discussed to improve the quality of the AVI data for books.

Looking at the recommendations of Grijze Jager part 5, the recommendations are a bit more diverse:



Figure 17: Grijze Jager recommendations in the app

As can be seen, other Grijze Jager books get recommended. It can also be seen that other fantasy books get recommended. However, when looking at the scores in the database, the first recommended book has a significantly higher score than the others:

| isbn13_book   | overall_score | tags_score | age_score | avi_score | remainder_score | sequence_number | isbn13_recommendation |
|---------------|---------------|------------|-----------|-----------|-----------------|-----------------|-----------------------|
| 9789025861346 | 0.7863        | 0.7242     | 1.0000    | 0.6445    | 1.0000          | 0               | 9789025744960         |
| 9789025861346 | 0.7863        | 0.7242     | 1.0000    | 0.6445    | 1.0000          | 0               | 9789025751944         |
| 9789025861346 | 0.6073        | 0.6402     | 0.5160    | 0.2985    | 0.4160          | 1               | 9789000374298         |
| 9789025861346 | 0.6047        | 0.6083     | 0.4756    | 0.2985    | 1.0000          | 2               | 97890085921660        |
| 9789025861346 | 0.5984        | 0.5876     | 0.5316    | 0.4439    | 1.0000          | 3               | 9789047713166         |
| 9789025861346 | 0.5977        | 0.6252     | 0.3605    | 0.6445    | 1.0000          | 4               | 9789020683547         |
| 9789025861346 | 0.5969        | 0.4799     | 1.0000    | 0.4439    | 1.0000          | 5               | 9789025752804         |
| 9789025861346 | 0.5839        | 0.5689     | 0.5316    | 0.6445    | 1.0000          | 6               | 9789047713173         |
| 9789025861346 | 0.5821        | 0.5697     | 0.6127    | 0.2929    | 0.6667          | 7               | 9781408855669         |
| 9789025861346 | 0.5787        | 0.5741     | 0.4784    | 0.3334    | 1.0000          | 8               | 9789054446415         |
| 9789025861346 | 0.5787        | 0.5741     | 0.4784    | 0.3334    | 1.0000          | 8               | 9789076174112         |
| 9789025861346 | 0.5787        | 0.5741     | 0.4784    | 0.3334    | 1.0000          | 8               | 9789076174129         |
| 9789025861346 | 0.5770        | 0.5985     | 0.3605    | 0.4439    | 1.0000          | 9               | 97890085922858        |
| 9789025861346 | 0.5765        | 0.6360     | 0.3285    | 1.0000    | 0.5227          | 10              | 9789025871642         |
| 9789025861346 | 0.5717        | 0.5658     | 0.4756    | 0.4439    | 1.0000          | 11              | 9789077826751         |
| 9789025861346 | 0.5717        | 0.5658     | 0.4756    | 0.4439    | 1.0000          | 11              | 97890085922025        |
| 9789025861346 | 0.5681        | 0.5763     | 0.5316    | 1.0000    | 0.5704          | 12              | 9789000374274         |
| 9789025861346 | 0.5661        | 0.5738     | 0.5316    | 0.2929    | 0.5681          | 13              | 9789025771508         |
| 9789025861346 | 0.5651        | 0.5445     | 0.5316    | 0.3334    | 1.0000          | 14              | 9789026156458         |
| 9789025861346 | 0.5642        | 0.6298     | 0.2929    | 0.3334    | 0.4966          | 15              | 9789025868000         |
| 9789025861346 | 0.5599        | 0.6066     | 0.3710    | 0.6445    | 0.4972          | 16              | 9789025868758         |
| 9789025861346 | 0.5591        | 0.6088     | 0.3576    | 1.0000    | 0.4930          | 17              | 9789025867621         |
| 9789025861346 | 0.5541        | 0.5399     | 0.6127    | 0.2929    | 0.5694          | 18              | 9789402707397         |
| 9789025861346 | 0.5528        | 0.5680     | 0.3576    | 1.0000    | 1.0000          | 19              | 9789056377144         |
| 9789025861346 | 0.5498        | 0.5335     | 0.4935    | 0.2929    | 1.0000          | 20              | 9789061697008         |
| 9789025861346 | 0.5498        | 0.5335     | 0.4935    | 0.2929    | 1.0000          | 20              | 9789061697015         |

Figure 18: Scores saved in database for Grijze Jager

The age and AVI scores are also a bit low, which can be explained by the fact that books like Grijze Jager and Harry Potter are not written specifically for children on elementary schools and are also targeting an older audience. This means that the recommendations are mainly based on the tags and remaining variables which can also be seen in the figure.

However, to obtain proper results to see if the recommendation system actually improves the reading pleasure for students, the system needs to be utilized for a few months before changes can be observed. At the moment of writing, a lot of children start with a book but do not finish it, either because they do not like the book or because it is too easy or difficult. If after a few months of utilizing the recommendation system the number of children that actually finish a book increases, only then would it be possible to state that the book recommendation system is working properly.

## 6 Future work

There is a lot of work that can and should be done in the future. Since scrapers are utilized, it is important to update them constantly as sites tend to change regularly. Also, because the recommendations are not perfect, possible improvements include:

1. Adding more variables to the recommendations array.
2. Adding new words to the keyword tree.
3. Updating weights of keywords in the tree.
4. Increasing the number of books for which a PDF file of the first few pages is known, which could be done by scraping them online or by going to libraries and scanning the books manually.
5. Utilizing AI. Right now, the weights for categories and keywords are based on trial and error. AI could probably determine optimal weights and hence improve the recommendation system even further.
6. Improving the quality of the AVI and age data about books. At the moment, these are based on what can be found on the internet. By utilizing the data known about students and the books they read, the data in the database can be overwritten by user driven data from the application to give a more accurate representation of the age and AVI level of the book.
7. Display the scores per category in the app to see why certain books are recommended. Right now the scores are saved in the database, but not made visible to the users. Displaying these scores can also enhance the recommendation system, because if a book gets recommended that does not closely resemble the other book, it is easier to identify where something should be changed in the recommendation system.

However, the most important thing that should be done is creating a second recommendation system that recommends books to users specifically. At the moment, item-item recommendations are generated based from one book to another. The idea of personal recommendations would be to generate a vector based on read and quit books by the user. This personal reading vector could then be used by ANNOY to find the best fitting books based on previously read books. The personal recommendations can also change from item-item to user-user recommendations. The generated personal reading vector is then not compared to books, but to other users with similar taste in order to recommend the best books based on previously read books. User-user recommendations are generally much better since it is hard to map all aspects of a book to precise data points as done in item-item recommendations. Unfortunately, more user data is required in order to achieve this result. Therefore the item-item recommendation system that has been built now is a solid foundation for future work.



## 7 Appendix

### A Preprocessing data

| Variable Name | Index | Value   |
|---------------|-------|---|
| Page count    | 0     | 1 if page count < 100   |
|               | 1     | 1 if $100 \leq$ page count < 250  |
|               | 2     | 1 if page count $\geq$ 250  |
| Book type     | 3     | 1 if Book type is 0 (reading book)  |
|               | 4     | 1 if Book type is 1 (informative book)  |
|               | 5     | 1 if Book type is 2 (picture book)  |
|               | 6     | 1 if Book type is 3 (comic book)  |
|               | 7     | 1 if Book type is 4 (poem collection)   |
|               | 8     | 1 if Book type is 5 (read aloud book, for the teacher to read to the class)       |
|               | 9     | 1 if Book type is 6 (book on orientation on reading)                              |
|               | 10    | 1 if Book type is 7 (search book, student needs to locate something in the book)  |
|               | 11    | 1 if Book type is 8 (pointing book, student needs to point at things in the book) |
| Book language | 12    | 1 if Book language is 1 (Dutch)   |
|               | 13    | 1 if Book language is 1 (English)   |
|               | 14    | 1 if Book language is 2 (German)  |
|               | 15    | 1 if Book language is 3 (French)  |
|               | 16    | 1 if Book language is 4 (Spanish)   |
|               | 17    | 1 if Book language is 5 (Frysian)   |
| Dyslexia      | 18    | 1 if book is a dyslexia version   |
| AVI           | 19    | 0.1 if AVI - 2 == 0   |
|               |       | 0.5 if AVI - 1 == 0   |
|               |       | 1 if AVI == 0   |
|               |       | 0.5 if AVI + 1 == 0   |
|               |       | 0.5 if AVI + 2 == 0   |
|               | 20    | 0.1 if AVI - 2 == 1   |
| AVI           | 20    | 0.5 if AVI - 1 == 1   |
|               |       | 1 if AVI == 1   |
|               |       | 0.5 if AVI + 1 == 1   |
|               |       | 0.5 if AVI + 2 == 1   |
|               |       | 0.1 if AVI - 2 == 2   |
|               | 21    | 0.5 if AVI - 1 == 2   |
| AVI           | 21    | 1 if AVI == 2   |
|               |       | 0.5 if AVI + 1 == 2   |
|               |       | 0.5 if AVI + 2 == 2   |
|               |       | 0.1 if AVI - 2 == 2   |
|               |       | 0.5 if AVI - 1 == 2   |
|               |       | 1 if AVI == 2   |

|    |  |
|----|--|
| 22 | 0.1 if $AVI - 2 == 3$<br>0.5 if $AVI - 1 == 3$<br>1 if $AVI == 3$<br>0.5 if $AVI + 1 == 3$<br>0.5 if $AVI + 2 == 3$      |
| 23 | 0.1 if $AVI - 2 == 4$<br>0.5 if $AVI - 1 == 4$<br>1 if $AVI == 4$<br>0.5 if $AVI + 1 == 4$<br>0.5 if $AVI + 2 == 4$      |
| 24 | 0.1 if $AVI - 2 == 5$<br>0.5 if $AVI - 1 == 5$<br>1 if $AVI == 5$<br>0.5 if $AVI + 1 == 5$<br>0.5 if $AVI + 2 == 5$      |
| 25 | 0.1 if $AVI - 2 == 6$<br>0.5 if $AVI - 1 == 6$<br>1 if $AVI == 6$<br>0.5 if $AVI + 1 == 6$<br>0.5 if $AVI + 2 == 6$      |
| 26 | 0.1 if $AVI - 2 == 7$<br>0.5 if $AVI - 1 == 7$<br>1 if $AVI == 7$<br>0.5 if $AVI + 1 == 7$<br>0.5 if $AVI + 2 == 7$      |
| 27 | 0.1 if $AVI - 2 == 8$<br>0.5 if $AVI - 1 == 8$<br>1 if $AVI == 8$<br>0.5 if $AVI + 1 == 8$<br>0.5 if $AVI + 2 == 8$      |
| 28 | 0.1 if $AVI - 2 == 9$<br>0.5 if $AVI - 1 == 9$<br>1 if $AVI == 9$<br>0.5 if $AVI + 1 == 9$<br>0.5 if $AVI + 2 == 9$      |
| 29 | 0.1 if $AVI - 2 == 10$<br>0.5 if $AVI - 1 == 10$<br>1 if $AVI == 10$<br>0.5 if $AVI + 1 == 10$<br>0.5 if $AVI + 2 == 10$ |
| 30 | 0.1 if $AVI - 2 == 11$<br>0.5 if $AVI - 1 == 11$<br>1 if $AVI == 11$<br>0.5 if $AVI + 1 == 11$<br>0.5 if $AVI + 2 == 11$ |

|                           |    |   |
|---------------------------|----|---|
| Words per page            | 31 | 1 if words per page < 50<br>0.25 if $50 \leq$ words per page < 100  |
|                           | 32 | 0.5 if words per page < 50<br>1 if $50 \leq$ words per page < 100<br>0.5 if $100 \leq$ words per page < 180   |
|                           | 33 | 0.5 if $50 \leq$ words per page < 100<br>1 if $100 \leq$ words per page < 180<br>0.5 if $180 \leq$ words per page < 300   |
|                           | 34 | 0.5 if $100 \leq$ words per page < 180<br>1 if $180 \leq$ words per page < 300<br>0.5 if $300 \leq$ words per page < 400  |
|                           | 35 | 0.5 if $180 \leq$ words per page < 300<br>1 if $300 \leq$ words per page < 400<br>0.75 if words per page $\geq$ 400   |
|                           | 36 | 0.5 if $300 \leq$ words per page < 400<br>1 if words per page $\geq$ 400  |
| Images per page           | 37 | 1 if images per page < 0.1  |
|                           | 38 | 1 if $0.1 \leq$ images per page < 0.67  |
|                           | 39 | 1 if images per page > 0.67   |
| Primary font size         | 40 | if (primary font size - average font size) > 1.5 then 1 if average font size $\leq$ 12<br>if (primary font size - average font size) $\leq$ 1.5 then 1 if primary font size $\leq$ 12 |
|                           | 41 | if (primary font size - average font size) > 1.5 then 1 if primary font size > 12<br>if (primary font size - average font size) $\leq$ 1.5 then 1 if primary font size > 12           |
| Percentage frequent words | 42 | The percentage/100  |
| Type/token ratio          | 43 | The type/token ratio divided by 100   |
| Release date              | 44 | 1 if release date < 2000<br>0.5 if $2000 \leq$ release date < 2005  |
|                           | 45 | 0.5 if release date < 2000<br>1 if $2000 \leq$ release date < 2005<br>0.5 if $2005 \leq$ release date < 2010  |
|                           | 46 | 0.5 if $2000 \leq$ release date < 2005<br>1 if $2005 \leq$ release date < 2010<br>0.5 if $2010 \leq$ release date < 2015  |
|                           | 47 | 0.5 if $2005 \leq$ release date < 2010<br>1 if $2010 \leq$ release date < 2015<br>0.5 if $2015 \leq$ release date < 2020  |
|                           | 48 | 0.5 if $2010 \leq$ release date < 2015<br>1 if $2015 \leq$ release date < 2020<br>0.5 if release date > 2020  |
|                           | 49 | 0.5 if $2015 \leq$ release date < 2020<br>1 if release date > 2020  |

|             |    |  |
|-------------|----|--|
| CILT-waarde | 50 | 1 if CILT-waarde < 54.9<br>0.5 if $54.9 \leq$ CILT-waarde < 56.9<br>0.25 if $56.9 \leq$ CILT-waarde < 58.9   |
|             | 51 | 0.5 if CILT-waarde < 54.9<br>1 if $56.9 \leq$ CILT-waarde < 58.9<br>0.5 if $58.9 \leq$ CILT-waarde < 61.9<br>0.25 if $61.9 \leq$ CILT-waarde < 63.9  |
|             | 52 | 0.1 if CILT-waarde < 54.9<br>0.5 if $54.9 \leq$ CILT-waarde < 56.9<br>1 if $56.9 \leq$ CILT-waarde < 58.9<br>0.5 if $58.9 \leq$ CILT-waarde < 61.9<br>0.25 if $61.9 \leq$ CILT-waarde < 63.9             |
|             | 53 | 0.1 if $54.9 \leq$ CILT-waarde < 56.9<br>0.5 if $56.9 \leq$ CILT-waarde < 58.9<br>1 if $58.9 \leq$ CILT-waarde < 61.9<br>0.5 if $61.9 \leq$ CILT-waarde < 63.9<br>0.25 if $63.9 \leq$ CILT-waarde < 65.9 |
|             | 54 | 0.1 if $56.9 \leq$ CILT-waarde < 58.9<br>0.5 if $58.9 \leq$ CILT-waarde < 61.9<br>1 if $61.9 \leq$ CILT-waarde < 63.9<br>0.5 if $63.9 \leq$ CILT-waarde < 65.9<br>0.25 if $65.9 \leq$ CILT-waarde < 67.9 |
|             | 55 | 0.1 if $58.9 \leq$ CILT-waarde < 61.9<br>0.5 if $61.9 \leq$ CILT-waarde < 63.9<br>1 if $63.9 \leq$ CILT-waarde < 65.9<br>0.5 if $65.9 \leq$ CILT-waarde < 67.9<br>0.25 if $67.9 \leq$ CILT-waarde < 69.9 |
|             | 56 | 0.1 if $61.9 \leq$ CILT-waarde < 63.9<br>0.5 if $63.9 \leq$ CILT-waarde < 65.9<br>1 if $65.9 \leq$ CILT-waarde < 67.9<br>0.5 if $67.9 \leq$ CILT-waarde < 69.9<br>0.25 if $69.9 \leq$ CILT-waarde < 71.9 |
|             | 57 | 0.1 if $63.9 \leq$ CILT-waarde < 65.9<br>0.5 if $65.9 \leq$ CILT-waarde < 67.9<br>1 if $67.9 \leq$ CILT-waarde < 69.9<br>0.5 if $69.9 \leq$ CILT-waarde < 71.9<br>0.25 if $71.9 \leq$ CILT-waarde < 73.9 |
|             | 58 | 0.1 if $65.9 \leq$ CILT-waarde < 67.9<br>0.5 if $67.9 \leq$ CILT-waarde < 69.9<br>1 if $69.9 \leq$ CILT-waarde < 71.9<br>0.5 if $71.9 \leq$ CILT-waarde < 73.9<br>0.25 if $73.9 \leq$ CILT-waarde < 74.9 |
|             | 59 | 0.1 if $67.9 \leq$ CILT-waarde < 69.9<br>0.5 if $69.9 \leq$ CILT-waarde < 71.9<br>1 if $71.9 \leq$ CILT-waarde < 73.9  |

|             |    |   |
|-------------|----|---|
|             |    | 0.5 if $73.9 \leq \text{CILT-waarde} < 74.9$<br>0.25 if $\text{CILT-waarde} \geq 74.9$  |
|             | 60 | 0.1 if $69.9 \leq \text{CILT-waarde} < 71.9$<br>0.5 if $71.9 \leq \text{CILT-waarde} < 73.9$<br>1 if $73.9 \leq \text{CILT-waarde} < 74.9$<br>0.5 if $\text{CILT-waarde} \geq 74.9$                                       |
|             | 61 | 0.1 if $71.9 \leq \text{CILT-waarde} < 73.9$<br>0.5 if $73.9 \leq \text{CILT-waarde} < 74.9$<br>1 if $\text{CILT-waarde} \geq 74.9$   |
| CLIB-waarde | 62 | 1 if $\text{CLIB-waarde} < -12$<br>0.5 if $-12 \leq \text{CLIB-waarde} < 7$<br>0.25 if $7 \leq \text{CLIB-waarde} < 20$   |
|             | 63 | 0.5 if $\text{CLIB-waarde} < -12$<br>1 if $7 \leq \text{CLIB-waarde} < 20$<br>0.5 if $20 \leq \text{CLIB-waarde} < 35$<br>0.25 if $35 \leq \text{CLIB-waarde} < 48$   |
|             | 64 | 0.1 if $\text{CLIB-waarde} < -12$<br>0.5 if $-12 \leq \text{CLIB-waarde} < 7$<br>1 if $7 \leq \text{CLIB-waarde} < 20$<br>0.5 if $20 \leq \text{CLIB-waarde} < 35$<br>0.25 if $35 \leq \text{CLIB-waarde} < 48$           |
|             | 65 | 0.1 if $-12 \leq \text{CLIB-waarde} < 7$<br>0.5 if $7 \leq \text{CLIB-waarde} < 20$<br>1 if $20 \leq \text{CLIB-waarde} < 35$<br>0.5 if $35 \leq \text{CLIB-waarde} < 48$<br>0.25 if $48 \leq \text{CLIB-waarde} < 61$    |
|             | 66 | 0.1 if $7 \leq \text{CLIB-waarde} < 20$<br>0.5 if $20 \leq \text{CLIB-waarde} < 35$<br>1 if $35 \leq \text{CLIB-waarde} < 48$<br>0.5 if $48 \leq \text{CLIB-waarde} < 61$<br>0.25 if $61 \leq \text{CLIB-waarde} < 74$    |
|             | 67 | 0.1 if $20 \leq \text{CLIB-waarde} < 35$<br>0.5 if $35 \leq \text{CLIB-waarde} < 48$<br>1 if $48 \leq \text{CLIB-waarde} < 61$<br>0.5 if $61 \leq \text{CLIB-waarde} < 74$<br>0.25 if $74 \leq \text{CLIB-waarde} < 69.9$ |
|             | 68 | 0.1 if $48 \leq \text{CLIB-waarde} < 61$<br>0.5 if $71.9 \leq \text{CLIB-waarde} < 73.9$<br>1 if $61 \leq \text{CLIB-waarde} < 74$<br>0.5 if $\text{CLIB-waarde} \geq 74$   |
|             | 69 | 0.1 if $48 \leq \text{CLIB-waarde} < 61$<br>0.5 if $61 \leq \text{CLIB-waarde} < 74$<br>1 if $\text{CLIB-waarde} \geq 74$   |

## B Code for generating random points in different dimensions

### B.1 Random points 1D

```
1 import random
2 import matplotlib.pyplot as plt
3
4 # Generate 100 random points in 1D space
5 points = [random.uniform(0, 1) for _ in range(100)]
6
7 # Count the number of points within 10% of the edges
8 count = sum(p <= 0.1 or p >= 0.9 for p in points)
9
10 # Get the points below 0.1 and above 0.9
11 below = [p for p in points if p < 0.1]
12 above = [p for p in points if p > 0.9]
13
14 # Plot the points
15 plt.scatter(points, [0]*100, c='b')
16 plt.scatter([0.1, 0.9], [0, 0], c='r')
17 plt.scatter(below, [0]*len(below), c='r')
18 plt.scatter(above, [0]*len(above), c='r')
19
20 # Add a title and show the plot
21 plt.title(f"Points within 10% of the edges: {count}")
22 plt.show()
```

## B.2 Random points 2D

```
1 import random
2 import matplotlib.pyplot as plt
3
4 # Generate 1000 random points in 2D space
5 points = [(random.uniform(0, 1), random.uniform(0, 1)) for _ in
6            range(1000)]
7
8 # Count the number of points within 10% of the edges
9 count = sum(min(p) <= 0.1 or min(p) >= 0.9 and max(p) <= 0.1 or max
10             (p) >= 0.9 for p in points)
11
12 # Get the points below 0.1 and above 0.9
13 below = [p for p in points if min(p) < 0.1 or max(p) < 0.1]
14 above = [p for p in points if min(p) > 0.9 or max(p) > 0.9]
15
16 # Plot the points
17 plt.scatter([p[0] for p in points], [p[1] for p in points], c='b')
18 plt.scatter([0.1, 0.9, 0.9, 0.1], [0.1, 0.1, 0.9, 0.9], c='r')
19 plt.scatter([p[0] for p in below], [p[1] for p in below], c='r')
20 plt.scatter([p[0] for p in above], [p[1] for p in above], c='r')
21
22 # Add a title and show the plot
23 plt.title(f"Points within 10% of the edges: {count}")
24 plt.show()
```

### B.3 Random points 3D

```
1 import random
2 import matplotlib.pyplot as plt
3
4 # Generate 1000 random points in 3D space
5 points = [(random.uniform(0, 1), random.uniform(0, 1), random.
6             uniform(0, 1)) for _ in range(1000)]
7
8 # Count the number of points within 10% of the edges
9 count = sum(1 for point in points if any(p <= 0.1 or p >= 0.9 for p
10      in point))
11
12 # Separate the points inside and outside the edges
13 edge_points = [point for point in points if any(p < 0.1 or p > 0.9
14      for p in point)]
15 area_points = [point for point in points if all(0.1 <= p <= 0.9 for
16      p in point)]
17
18 # Plot the points
19 fig = plt.figure()
20 ax = fig.add_subplot(111, projection='3d')
21 ax.scatter([p[0] for p in edge_points], [p[1] for p in edge_points
22      ], [p[2] for p in edge_points], c='r')
23 ax.scatter([p[0] for p in area_points], [p[1] for p in area_points
24      ], [p[2] for p in area_points], c='b')
25
26 # Add edges to the plot
27 ax.plot([0.1, 0.1, 0.1, 0.1, 0.9, 0.9, 0.9, 0.9],
28         [0.1, 0.1, 0.9, 0.9, 0.1, 0.1, 0.9, 0.9],
29         [0.1, 0.9, 0.1, 0.9, 0.1, 0.9, 0.1, 0.9], c='g')
30
31 # Add a title and show the plot
32 plt.title(f"Points within 10% of the edges: {count}")
33 plt.show()
```



## C Euclidean vs Angular vs Manhattan vs Dot product distance

```
1 import matplotlib.pyplot as plt
2 from matplotlib.patches import FancyArrowPatch, Arc, Circle, Patch
3 import numpy as np
4 import math
5
6 # Create a 2D plot
7 fig, ax = plt.subplots()
8
9 # Define the vectors
10 v1 = np.array([1, 4])
11 v2 = np.array([8, 8])
12 v3 = np.array([3, 1])
13
14 # Plot the vectors
15 ax.arrow(0, 0, v1[0], v1[1], head_width=0.5, head_length=0.5, color='blue')
16 ax.arrow(0, 0, v2[0], v2[1], head_width=0.5, head_length=0.5, color='blue')
17 ax.arrow(0, 0, v3[0], v3[1], head_width=0.5, head_length=0.5, color='blue')
18 ax.text(v1[0], v1[1]+0.8, 'v1', ha='center', va='center', color='blue', fontsize=12)
19 ax.text(v2[0], v2[1]+0.8, 'v2', ha='center', va='center', color='blue', fontsize=12)
20 ax.text(v3[0]+0.2, v3[1]-0.6, 'v3', ha='center', va='center', color='blue', fontsize=12)
21
22 # Plot the line and display the euclidean distance
23 d = np.linalg.norm(v2 - v1)
24 ax.plot([v1[0], v2[0]], [v1[1], v2[1]], color='red')
25 ax.text(4, 6, f'{d:.2f}', ha='center', color='red')
26 d2 = np.linalg.norm(v3 - v1)
27 ax.plot([v1[0], v3[0]], [v1[1], v3[1]], color='red')
28 ax.text(v3[0]-0.3, v3[1]+1, f'{d2:.2f}', ha='center', color='red')
29
30 # Plot the line and display the manhattan distance
31 ax.plot([v1[0], v2[0]], [v2[1], v2[1]], color='purple')
32 ax.plot([v1[0], v1[0]], [v2[1], v1[1]], color='purple')
33 ax.text(4.5, 8.2, f'{7}', ha='center', color='purple')
34 ax.text(0.7, 6.5, f'{4}', ha='center', color='purple')
35 ax.text(1, 8.2, '4+7 = 11', ha='center', color='purple')
36 ax.plot([v1[0], v3[0]], [v1[1], v1[1]], color='purple')
37 ax.plot([v3[0], v3[0]], [v1[1], v3[1]], color='purple')
38 ax.text(2.5, 4.2, f'{2}', ha='center', color='purple')
39 ax.text(3.2, 2, f'{3}', ha='center', color='purple')
40 ax.text(3.5, 4.2, '2+3 = 5', ha='center', color='purple')
41
42 # Calculate the angle between the vectors
43 cos_dis = 1 - np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))
44 angle = np.arccos(np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))) * 180 / np.pi
45 rad_angle = math.radians(angle)
```

```

46 cos_dis2 = 1 - np.dot(v1, v3) / (np.linalg.norm(v1) * np.linalg.
    norm(v3))
47 angle2 = np.arccos(np.dot(v1, v3) / (np.linalg.norm(v1) * np.linalg.
    .norm(v3))) * 180 / np.pi
48 rad_angle2 = math.radians(angle2)
49
50 # Plot the arc and display the angle in degrees and radians and
    cosine similarity
51 center = (0, 0)
52 arc = Arc(center, 5.5, 5.5, angle=0, theta1=45, theta2=math.degrees
    (math.atan(4/1)), color='green', linewidth=2)
53 arc2 = Arc(center, 5.4, 5.4, angle=0, theta1=45, theta2=math.
    degrees(math.atan(4/1)), color='orange', linewidth=2)
54 arc3 = Arc(center, 2, 2, angle=0, theta1=math.degrees(math.atan
    (1/3)), theta2=math.degrees(math.atan(4/1)), color='green',
    linewidth=2)
55 arc4 = Arc(center, 1.9, 1.9, angle=0, theta1=math.degrees(math.atan
    (1/3)), theta2=math.degrees(math.atan(4/1)), color='orange',
    linewidth=2)
56 ax.add_patch(arc)
57 ax.add_patch(arc2)
58 ax.add_patch(arc3)
59 ax.add_patch(arc4)
60 ax.text(0.3, 2.8, f'{cos_dis:.2f}', ha='center', va='center', color
    ='orange')
61 ax.text(1.7, 3.3, f'{rad_angle:.2f} rad', ha='center', va='center',
    color='green')
62 ax.text(1.5, 2.8, f'{angle:.2f}', ha='center', va='center', color='
    green')
63 ax.text(1.2, 1, f'{rad_angle2:.2f} rad', ha='center', va='center',
    color='green')
64 ax.text(1.1, 1.5, f'{angle2:.2f}', ha='center', va='center', color=
    'green')
65 ax.text(1.5, 0.2, f'{cos_dis2:.2f}', ha='center', va='center',
    color='orange')
66
67 # Set the axis limits and labels
68 ax.set_xlim([0, 10])
69 ax.set_ylim([0, 10])
70 ax.set_xlabel('X')
71 ax.set_ylabel('Y')
72
73 # create a custom legend with the colors used in the plot
74 legend_elements = [Patch(facecolor='blue', edgecolor='blue', label=
    'Vector'),
75                     Patch(facecolor='green', edgecolor='green',
    label='Angle'),
76                     Patch(facecolor='red', edgecolor='red', label='
    Euclidean'),
77                     Patch(facecolor='purple', edgecolor='purple',
    label='Manhattan'),
78                     Patch(facecolor='orange', edgecolor='orange',
    label='Dot product')]
79 ax.legend(handles=legend_elements, loc='lower right')
80
81 # Show the plot
82 plt.show()

```

## D Code for limit/inside ratio

### D.1 1D

```
1 import matplotlib.pyplot as plt
2 import matplotlib.patches as patches
3
4 # Plot the points
5 L = 1
6 fig, ax = plt.subplots()
7 ax.plot([0,1], [0,0], color='blue', linewidth=2)
8 ax.scatter(0, 0, color='red', linewidth=2)
9 ax.scatter(1, 0, color='red', linewidth=2)
10
11 # set the tick labels for the x and y-axis
12 x_ticks = [0, 0.25*L, 0.5*L, 0.75*L, L]
13 x_ticklabels = ['0', '0.25L', '0.5L', '0.75L', 'L']
14 ax.set_xticks(x_ticks)
15 ax.set_xticklabels(x_ticklabels)
16
17 y_ticks = [0, 0.25*L, 0.5*L, 0.75*L, L]
18 y_ticklabels = ['0', '0.25L', '0.5L', '0.75L', 'L']
19 ax.set_yticks(y_ticks)
20 ax.set_yticklabels(y_ticklabels)
21
22 # Add a title and show the plot
23 plt.title("Limit/inside ratio")
24 plt.show()
```

## D.2 2D

```
1 import matplotlib.pyplot as plt
2 import matplotlib.patches as patches
3
4 # Plot the points
5 L = 1
6 rect = patches.Rectangle((0, 0), L, L, facecolor='blue', edgecolor
7                             = 'red', linewidth=5)
8 fig, ax = plt.subplots()
9 ax.add_patch(rect)
10
11 # set the tick labels for the x and y-axis
12 x_ticks = [0, 0.25*L, 0.5*L, 0.75*L, L]
13 x_ticklabels = ['0', '0.25L', '0.5L', '0.75L', 'L']
14 ax.set_xticks(x_ticks)
15 ax.set_xticklabels(x_ticklabels)
16
17 y_ticks = [0, 0.25*L, 0.5*L, 0.75*L, L]
18 y_ticklabels = ['0', '0.25L', '0.5L', '0.75L', 'L']
19 ax.set_yticks(y_ticks)
20 ax.set_yticklabels(y_ticklabels)
21
22 # Add a title and show the plot
23 plt.title("Limit/inside ratio")
24 plt.show()
```

### D.3 3D

```
1 import matplotlib.pyplot as plt
2 from mpl_toolkits.mplot3d.art3d import Poly3DCollection
3 import numpy as np
4
5 # Define cube vertices
6 L = 1
7 verts = np.array([(0, 0, 0), (0, L, 0), (L, L, 0), (L, 0, 0),
8                  (0, 0, L), (0, L, L), (L, L, L), (L, 0, L)])
9
10 # Define cube faces
11 faces = np.array([(0,1,2,3), (0,4,5,1), (1,5,6,2),
12                  (2,6,7,3), (3,7,4,0), (4,7,6,5)])
13
14 # Create plot and add cube
15 fig = plt.figure()
16 ax = fig.add_subplot(111, projection='3d')
17 cube = Poly3DCollection(
18     [verts[faces[i]] for i in range(len(faces))],
19     edgecolors='red',
20     facecolors='blue',
21     linewidth=3)
22 ax.add_collection3d(cube)
23
24 # Set plot limits and labels
25 ax.set_xlim([0, L])
26 ax.set_ylim([0, L])
27 ax.set_zlim([0, L])
28 ax.set_xlabel('X')
29 ax.set_ylabel('Y')
30 ax.set_zlabel('Z')
31
32 plt.title("Limit/inside ratio")
33
34 plt.show()
```

## References

- [1] PISA. *Leesvaardigheid Nederlandse scholieren daalt*. [https://www.pisa-nederland.nl/resultaten2018/#sect\\_leesvaardigheid](https://www.pisa-nederland.nl/resultaten2018/#sect_leesvaardigheid).
- [2] Stijn Dautzenberg and Tom Buunk. *Forse daling van leesplezier en leesvaardigheid*. <https://onderwijsinbeeld.nl/over-ons/>. 2020.
- [3] Maarten Beernink. *Titelbank: de schatkamer van alle boektitels*. <https://www.hetboekenschap.nl/titelbank/>. Apr. 2016.
- [4] Titelbank. [https://www.titelbank.nl/pls/ttb/f?p=103:4010:::N0:RP::&cs=3v\\_kd9QLC2Au0iBfbKrhtFQnmWmv0PcXzvZRGR5deT6UkXths8HkpRB2PI9iDnm945qkBS\\_HPeSjSIKcqAAeS1w](https://www.titelbank.nl/pls/ttb/f?p=103:4010:::N0:RP::&cs=3v_kd9QLC2Au0iBfbKrhtFQnmWmv0PcXzvZRGR5deT6UkXths8HkpRB2PI9iDnm945qkBS_HPeSjSIKcqAAeS1w).
- [5] Sam Ingalls. *What Is an FTP Server and How Does It Work?* <https://www.serverwatch.com/guides/ftp-server/>. Aug. 2021.
- [6] Python Software Foundation. *Python Documentation: ftplib*. 2023. URL: <https://docs.python.org/3/library/ftplib.html>.
- [7] Kenneth Reitz and Contributors. *Requests: HTTP for Humans*. <https://pypi.org/project/requests/>. 2022.
- [8] Leonard Richardson and Beautiful Soup contributors. *Beautiful Soup: HTML parsing for Python*. <https://pypi.org/project/beautifulsoup4/>. 2022.
- [9] Rosemary Slagmolen. *Dit boek is lekker makkelijk!* 2008. URL: <https://studenttheses.uu.nl/handle/20.500.12932/9856>.
- [10] Marc van Oostendorp. *De tien keer honderd meest gebruikte woorden van de taal*. May 2016. URL: <https://neerlandistiek.nl/2016/05/de-tien-keer-honderd-meest-gebruikte-woorden-van-de-taal/>.
- [11] Yusuke Shinyama. *PDFMiner*. <https://pypi.org/project/pdfminer/>. 2021.
- [12] user2314737. *How does one obtain the location of text in a PDF with pdfminer?* <https://stackoverflow.com/questions/25248140/how-does-one-obtain-the-location-of-text-in-a-pdf-with-pdfminer>. 2014.
- [13] *Meest-gebruikte.nl: Meest gebruikte afkortingen*. <https://www.meest-gebruikte.nl/meest-gebruikte-afkortingen/>.
- [14] Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Toolkit*. <https://www.nltk.org/>. 2009.
- [15] SystemDesign. *System Design Interview: Recommendation System Design (As Used By YouTube, Netflix etc.)* <https://medium.com/double-pointer/system-design-interview-recommendation-system-design-as-used-by-youtube-netflix-etc-c457aaec3ab>. Sept. 2021.
- [16] Social Data Science Deep Learning Portfolio. *Approximate Nearest Neighbors Oh Yeah (ANNOY)*. <https://sds-aau.github.io/M3Port19/portfolio/ann/>.

- [17] Erik Bernhardsson. *Annoy: Approximate Nearest Neighbors in C++/Python*. <https://github.com/spotify/annoy>. 2021.
- [18] Erik Bernhardsson. *annoy*. <https://pypi.org/project/annoy/>. 2021.
- [19] James Thorn. *The Surprising Behaviour of Distance Metrics in High Dimensions*. <https://towardsdatascience.com/the-surprising-behaviour-of-distance-metrics-in-high-dimensions-c2cb72779ea6>. 2021.
- [20] Anuj Shrivastav. *Curse Of Dimensionality: The curse that all ML Engineers need to deal with*. <https://medium.com/analytics-vidhya/curse-of-dimensionality-the-curse-that-all-ml-engineers-need-to-deal-with-5d459d39dc8a>. Feb. 2020.
- [21] Shuyin Xia, Zhongyang Xiong, Yueguo Luo, Wei Xu, Guanghua Zhang. *Effectiveness of the Euclidean distance in high dimensional spaces*. <https://www.sciencedirect.com/science/article/abs/pii/S0030402615011493>. 2015.
- [22] Python Software Foundation. *Enum: Support for enumerations*. <https://docs.python.org/3/library/enum.html>. 2021.