AALBORG UNIVERSITY

STUDENT REPORT

Aalborg University Copenhagen
A.C. Meyers Vænge 15
2450 København SV
Semester Coordinator: Henning Olesen
Secretary: Maiken Keller

Title:
AAU student report recommendation system

Theme:
Service development

Project Period:
Spring Semester 2020

Project Group:
N/A

Participant(s):
Sebastian Clausen

Supervisor(s):
Sokol Kosta

Pages: 78

Date of Completion:
August 6, 2020

Abstract:

Aalborg University has a Digital Project Library where students upload comprehensive project documentation from their studies, including bachelor's theses, master's theses and regular semester papers. There are more than 40 thousand papers in the library dating back more than 20 years, detailing academic work of analysis and applied theory in all sorts of domains. They lie mostly unused and therefore, this project explores a way to encourage more students to read and get inspired by these papers. Using an information retrieval and recommendation system hybrid, a system is built based on user testing, which can recommend papers to students, using content-based filtering. The use of university courses as inputs to the system is tested, showing some promise to a convenient way of indicating interest. The system employs a microservice architecture, tailored to the nature of recommendation systems, where trends change quickly and failure to improve is bad for business. A prototype is developed which showcase the recommendation of student papers, as well as the augmentation of papers using several additional microservices.

AAU student report recommendation system

by Sebastian Clausen

Contents

1 Introduction

Millions of hours have been spent working on and documenting tens of thousand of student projects at Aalborg University. All of these hours culminate in nice comprehensive reports and a final exams at the end of the semester, after which everyone moves on with their lives. Nearly every single time, that exam is the last time anyone cares about what was written in the report. All the tedious research into state-of-the-art concepts and ideas, the analyses and creative work that takes those ideas and apply or build upon them, to create new knowledge in fascinating spaces and under unique practical circumstances - all of that, laid to rest in unmarked graves. In truth, they are still there, yet most of the time they are invisible unless you are actively searching for it. This project aims to raise these nuggets of wisdom to the surface, where they can be exposed to an audience of university students seeking education, inspiration or recreation. Aalborg University has collected theses and student projects from all their students since 2008 (according to the official site, however some papers date back even further) and made them available on Aalborg University Digital Project Library, and this is where student papers are buried, never gone, but forgotten. Thus, this is the opposite of an eulogy for the demise of insight, this is its revival and second chance. There is a long way from what the Digital Project Library is today, to what it could and presumably was intended to do, i.e. being a place for students to go and get inspired about possible directions and applications for what they are being taught in their classes and what is going on in the world of academia. The purpose of this project is to facilitate that inspirational journey, in a way that is convenient to students, by demanding little effort from them and effortlessly accurate in assuming their interest. This solution is initially inspired by recommendation systems, which represents a large area of study where inputs comes in all shapes and forms, and are implicit and explicit. Providing students with a service that not only is easy to use, but also effective in its purpose will hopefully encourage usage of the project library. There are multiple facets to this design challenge, as it involves users heavily and also has advanced technical demands. The service design challenge lies in catering to users' different interests and focusing on their interaction with the system. Thus, working with users during development to establish a good user experience is important to ensure that it is enticing to use and continue using for as long as it can provide a value. That includes a user-centric requirements gathering, where choices are based on what users find valuable to the system. The technical aspect to this system concerns the architectural choices and the development process. The system's innate dependence on something as complex as human interest, effectively means it must be mutable to an extent where all parts are exchangeable in favor of another part, which might or might not improve the system. Importance lies in the rapidness of the exchange and ability to try new things without large development costs. Such modularity has to be paired with a methodology, which focuses on the life cycle and the stages of analysis, design, development, testing, implementation, documentation and evaluation.

2 Problem analysis

The project library of Aalborg University contains thousands upon thousands of comprehensive academic papers, written by students of Aalborg university during their studies. The university practices the Problem-based learning model, which encourages making relations between theory and real-life problems or circumstances. For many of the students, this relation is explored anew in multiple-month long project each semester, with different topics related to the students' courses at that time and the project is extensively documented in a final paper. Following the final delivery of each semester project's paper, it is uploaded to the project library to... well... why?
The purpose of the project library, beyond showing off knowledge produced by AAU students in their theses, is unclear. The theses are publicly available and showcase the talent of AAU students, however every other paper in the library requires AAU login access, which limits them from the public. Thus, since they are not intended for the public, they must be intended for students and faculty. Consider the general purpose of a library, which most often is to educate and entertain. Considering an AAU student, they might be subject to both purposes. Reading about academic ventures and discoveries could be both educational and entertaining, but there is another level of education in these papers. The fact that the students themselves will have to write papers in a similar fashion, makes the papers on the library useful as an inspirational resource for content and especially for format. This is a way to learn by example. These reports are finished products that demonstrate how an academic paper could look. Content-wise they show how theory learned during the education can be applied to problems, format-wise they show combinations of chapters to include and their chronology, citation and reference techniques, and formalities such as page layout and composition.

2.1 Search tools

The actual research, analysis and discoveries are not that relevant, when looking at the papers with inspiration for a current project as the goal. When the content is not relevant, it is not because every project is so unique that no other project discuss similar topics, it is because the papers that are in fact relevant, are not immediately available when there is a need for them. The Digital Project Library's attempt at advertising the papers, include browsing through educations and text searches, with a set of filters. The "advanced search" page of the library is show in Figure 1. The search seemingly finds all papers that contain one of the search terms and lists all of them by date and then alphabetically for each year. E.g. for the year 2020, there are papers sorted from A-Z and for the year 2019, there is another set of papers sorted A-Z. It has no regard for how well the papers match, meaning that unless the search terms are very unique words, it is entirely impossible to find the papers that actually concern the search query, rather than including just an off-hand mention of it. With so much information available, but only these limited tools to retrieve it, users can easily become overwhelmed and give up on their

search.

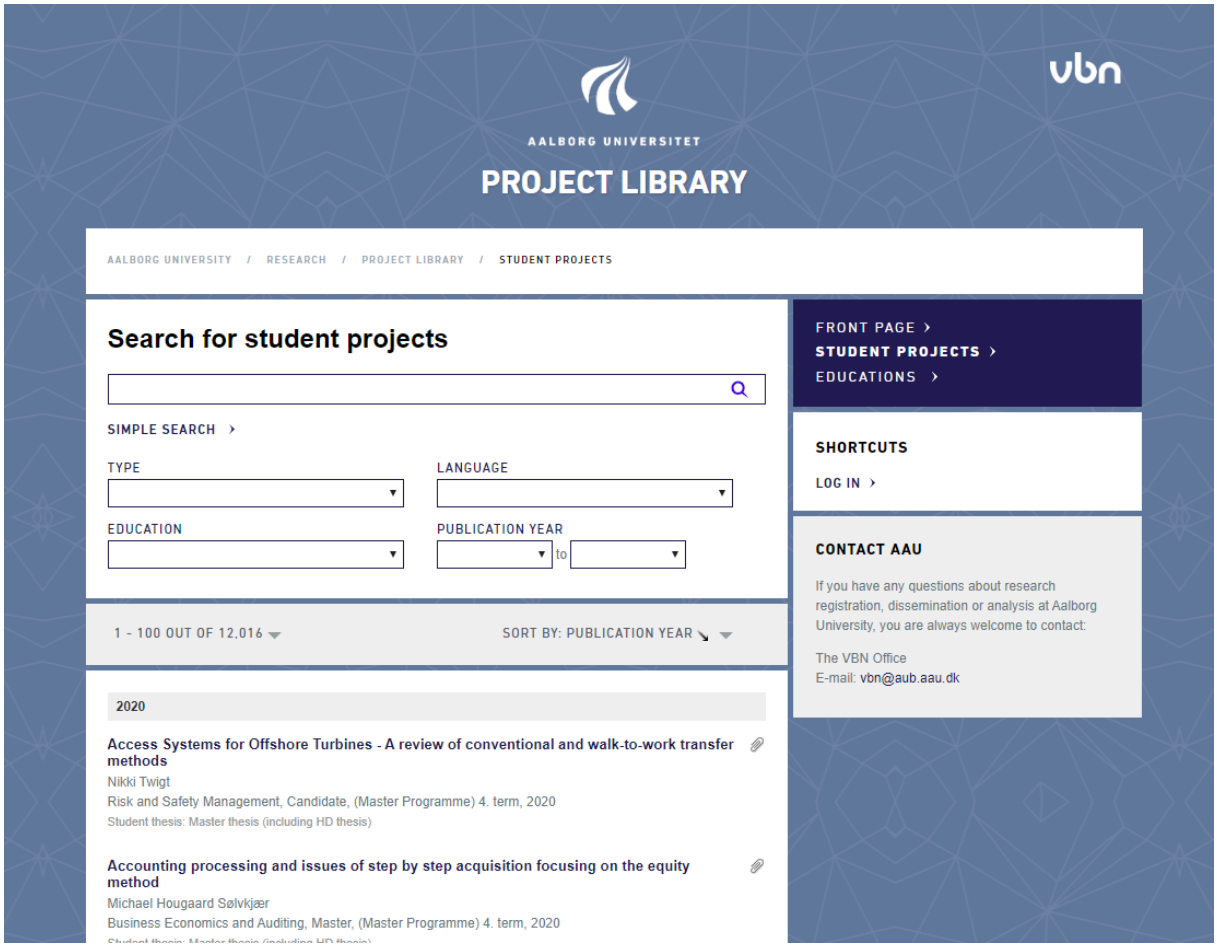

Figure 1: Project library advanced search

2.2 Paper diversity

The student papers are produced by students from the first semester of university until the last, which means there are likely a significant disparity of quality between the reports. The disparity will not only show itself in the academic content of the papers, which hopefully improves as the students' learn and practice, but it will also show in the format of the papers, as writing proper academic papers is also a skill that has to be learned and practiced. Structure of report chapters will vary and elements such as title pages, tables of content and abstracts will usually exist in all the papers, but how it is laid out is also something that varies. The differences stem not only from inexperienced students, but also seemingly is affected by the culture of the educations in the study boards. Perhaps, the student are instructed by their teachers or supervisors to write a certain way, or perhaps they see how published academic papers in their fields are formed. Some are very particularly of how the report should be shaped, while others are more lenient and allow a variety of different formats. Some studies

on the extreme ends of this, are Computer Science and Urban Design. Papers by students on the Master's level Computer Science study line tend to be much more homogeneous compared to papers by students on the Master's level Urban Design study line. The explanation for this is probably found in the guidelines set for the students. Computer Science students might be instructed to follow a set of directions for paper format, but Urban Design students would have more liberty with regards to format, allowing them to be more creative and unique in their project documentations. An example of this disparity is the labeling of chapters. A large majority of Computer Science papers have clearly defined and numbered sections, but Urban Design papers focus more on the visuals, meaning the sections might be clearly defined, but using design and aesthetic tools, such as constraining a section to a single page, and fitting as much into that page as possible, with no regard for a coherence to the rest of the paper's format. Commonly seen, is the usage of multiple columns and adjustment of column widths and number. Computer Science papers occasionally have multiple columns, but it is always two, and always stays consistent through the paper. Further, the visual approach often result in graphics, i.e. pictures and drawings, as a trade-off for text. Format is one factor which is probable to have a high diversity, with large distances between the two ends of the spectrum and it follows that widespread significant diversity leads to a heterogeneous corpus. The medium is the same, but the format, contents and sizes are not. There are a few other immediately obvious factors that lead to a heterogeneous corpus besides the semantic content, which include the written language and the length of the paper. The corpus is multilingual, because some educations are taught in English and others in Danish, thus the corresponding papers by students adhering to these educations, write in the appropriate language for the education. There are also educations centered around other nations and languages, which result in more possible languages for a paper, either wholly or mixed, which might have to be accounted for, e.g. Spanish or German. Interests do not necessarily adhere to a language, but the way interests are usually expressed in information search is. Therefore, discrepancy in language of a topic expression and the language in which a potential document is written, could cause severe issues. This would be a familiar issue to many, because it exists even with the very popular search engines like Google, where the results to a search in a particular language generate results in that same language, despite results in another language being more topically relevant. That is not to say Google does not have the capability to translate between languages, because they are obviously very capable as a leading provider of translation services, but at the default state they choose not to. It must be assumed that the world's leading search engine, do their utmost to show the users what they are looking for, and Google's decision to emphasize on multilingual search results, speaks to the users' desire for these. Another factor, length of the papers, dictate the volume of available data, as longer papers will have more semantic data available. That is not to say that more information is always better or even that much information is necessary, but it does matter in terms of semantic analysis, as it cares about which words are used and how often. It is inherently challenging to process heterogeneous

types, as it might require different and additional processing techniques for some papers, to achieve a comparable result. And that is assuming it is even possible to arrive at something which can be generalized for all documents, because it could be necessary to compromise with the lowest common denominator, in order to include papers that do not have as much data available as others or data which is different and cannot be converted in to something useful.

There could be instances, where a paper does not have sufficient data available, for it to contribute any value to the system, at which point it does not make sense to go to the lowest common denominator, but rather it might result in complete exclusion of that paper.

2.3 Library statistics

The papers on the project library are sorted as seen in Figure 2. The departments of the university are each listed with their underlying study boards and the study boards contain underlying educations. Figure 2 shows the structure of the project library, which is presented to the user. The underlying

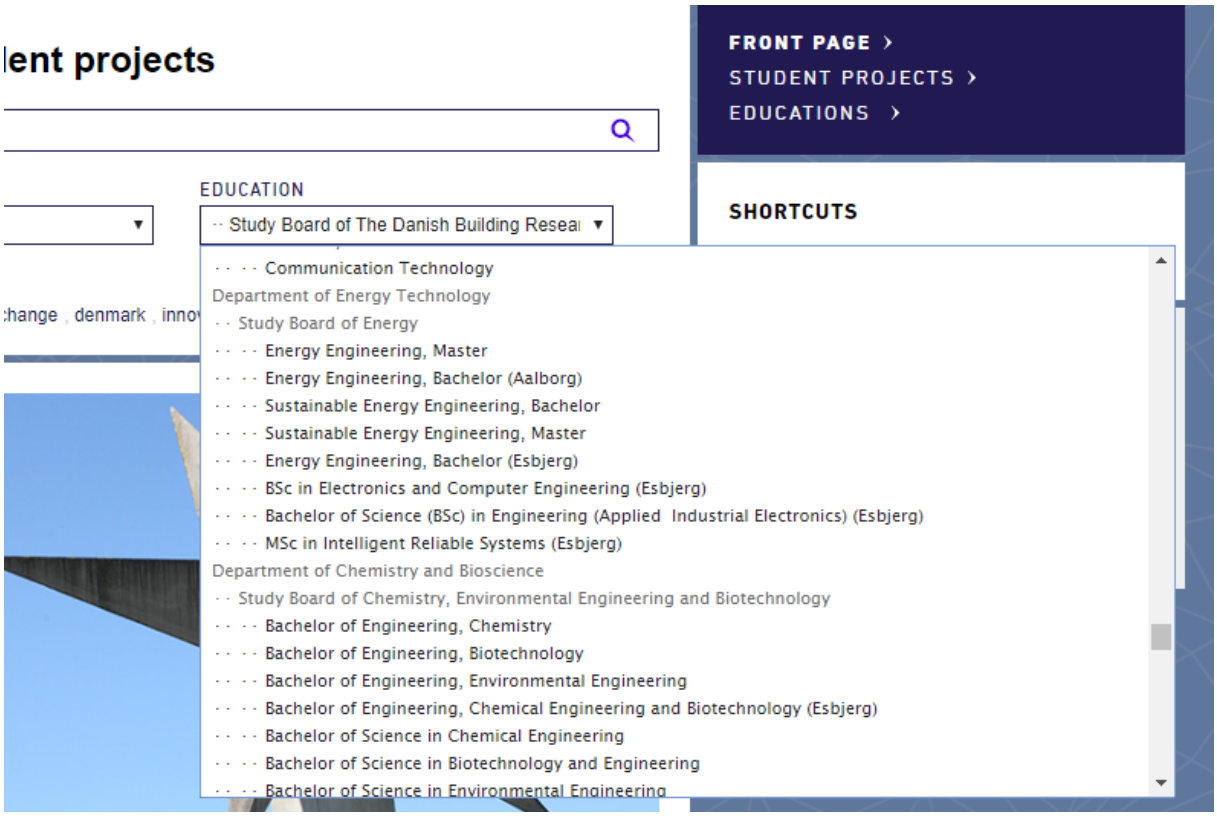

Figure 2: Project library department and study board hierarchy

educations are particular specialties within the study board, but the educations belonging to a study board often have overlapping courses among them and students from a particular education do not necessarily have exactly the same courses either. Thus, it seems there should be speculation whether

to group papers by study board, rather than by educations. This way all potential courses of a student on any of the educations is included now and in the future, where it educations are more likely to change than study boards. However, it seems that some study boards are significantly larger than others and some have significantly more educations within them. It must be assumed that educations are created to teach specific things, thus making them somewhat different from each other. Therefore, it follows that more educations within a study board, will bring more diversity to the study board. Besides the variation of topics within a study board, educations also vary with regard to approaches and languages, and suddenly it seems as if the potential paper diversity is rather high. Instead of leaving it to speculation, analyses have been made, to get an idea of to which degree these potential diversity factors actually exist and how they impact the corpus. Figure 3 shows a distribution of student papers across study boards. This distribution gives an idea of how many papers pertain to each of the study boards and accordingly, to the academic topics those study boards concerns themselves with (see 11.4.1 for a detailed list of this distribution). The largest disparity, which is apparent in the figure, shows a fairly even distribution in the range zero to approximately 3600. The study board with the most papers is the study board of computer science with 3611, which makes up approximately 8,5% of the total 41785 papers. That is a significant portion compared to the more than half (26 out of 44 study boards) of all study boards that have less than 1000 papers in their study board, with one as few as 28. Further, within the top five study boards with the highest number of papers, there are two other study boards than that of computer science, which are IT related, namely the study board of electronics and information technology and communication and digital media. These three educations alone comprise more than a fifth of the papers (22%), majorly steering the general corpus domain towards IT. Figure 4 shows a distribution of the number of educations within each study board (see

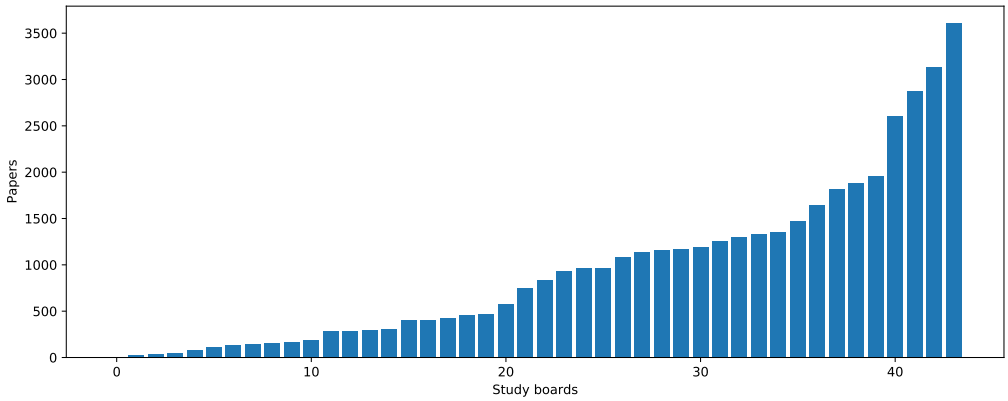

Figure 3: Papers by study board. Total: 41785

section 11.4.1 for a detailed list of this distribution). Electronics and information technology is the study board with the most educations at 61 (See Table 1) and presumably also the study board with

Board	Papers	Educations
Study Board of Computer Science	3611	28
Study Board of Planning and Surveying	3135	21
Study Board of Electronics and Information Technology	2881	61
Study Board of Communication and Digital Media	2601	19
Study Board of Materials and Production	1962	30
Study Board of Civil Engineering	1886	27
Study Board of Architecture and Design	1820	8
Study Board of Business Economics	1645	8
Study Board of Media Technology	1474	6
Study Board of Techno-Anthropology and Sustainable Design	1355	6

Table 1: Highest paper count study boards

the most diversity within. It is inaccurate to say as a blanket statement that number of educations is directly proportional to level of diversity, since it also depends on the particular study board and how it has been defined. The study board of music therapy concerns a rather specific and fairly narrow field, where as the study board of electronics and information technology concerns a broad field, which is not very strictly defined.

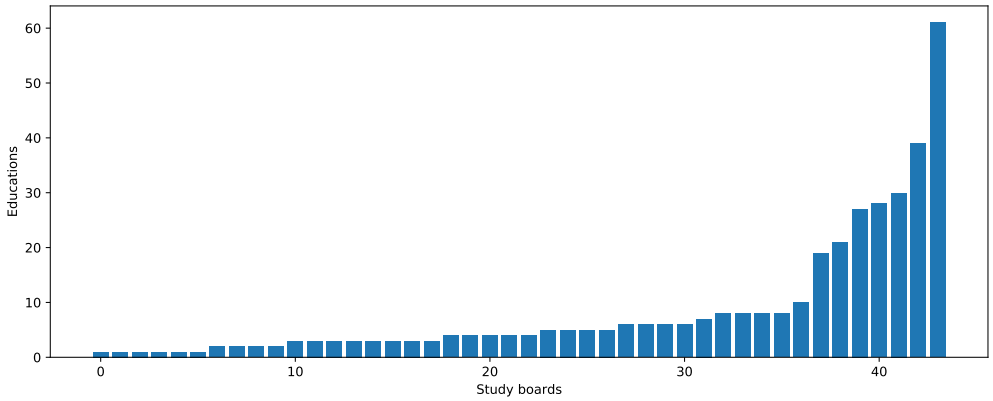

Figure 4: Educations by study board. Total: 376

2.3.1 Content analysis

Content analysis is performed on the papers, to estimate some properties of the documents that give a better understanding of the documents. Here focus is on the diversity factors of language and size of the papers within the different study boards. Analyzing the content is more challenging than the previous analyses of corpus, which did not care about the words and sentences. In this section there will be some caveats regarding the number of included papers, due to aforementioned variety of formats (see sections 2.2, 2.4).

Language

The papers in the project library are written in different languages. There are six languages in use: Danish, English, Spanish, German, Norwegian and Swedish. All the reports are analyzed to try and find out which language is dominant, thus being able to estimate the distribution of languages across the library. To avoid doing this manually, the tool "langdetect" [1] was used to detect the language of all the machine readable papers in the library. It can detect 55 languages based on a textual input of arbitrary length - with longer inputs being more reliable. It assigns a probability to each language for the input, the highest probability is considered to be the language. Some reports have mixed language, e.g. most parts are in English, but there are summaries, interviews, appendices etc. that are in Danish. Therefore, using the entire paper is confusing to the detector, and instead five samples are taken from the report. The samples are taken throughout the main portion of the report, to avoid aforementioned summaries and appendices, that are more likely to be in a language other than the main content. Each sample is 100 characters and is run through the detector. The language that comes up the most of the five samples, is selected.

A few study boards have been selected to demonstrate how this tool performs. The tool's results are compared to a manual count of the labels set by authors for the papers. The boards are the study boards of psychology, international affairs and electronics and information technology. Shown in Figure 5 is the distribution of languages of the papers within those study boards, where the the groups are Danish (da), English (en) and the remaining languages of Spanish, German, Norwegian and Swedish (other). The bars are marked with automatic (A) or manual (M), depending on how they were counted.

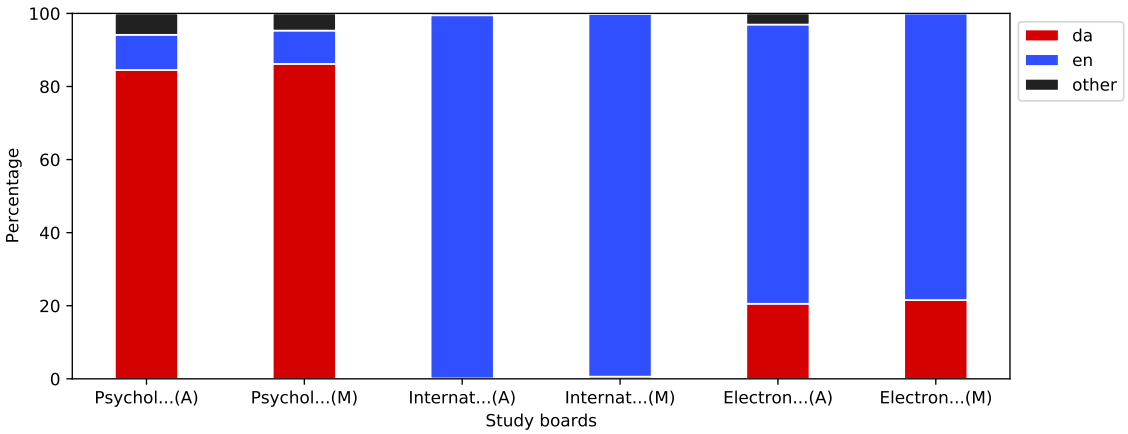

Figure 5: Language distribution of study boards of psychology, international affairs and electronics and information technology by automatic detection (A) or manual count (M)

The observed distributions of the study board of psychology in Figure 5, indicate that approximately 85% of papers are in Danish and the remaining 15% are split among English, Norwegian and Swedish. This shows a common trend among study boards, where they are predominantly English or

	da	en	other
Papers	21550 (52.0%)	19108 (46.0%)	814 (2.0%)

Table 2: Language distributions across all papers

Danish, but still have fractions of other languages to varying degrees. Although it is difficult to spot in Figure 5 the international affairs' study board bar show that automatic detection is more accurate in some case. The 0.6% (ten papers) that have the Danish label on the project library under the study board of international affairs are all actually written in English, and have been mislabeled by the authors. However, on the other side of this, the automatic detection predicted some language labels that were really wrong, e.g. Indonesian, Estonian, which are non-existent on the library, and sometimes even had trouble finding a label for a paper. Neither solution is perfect, however the automatic version saves a lot of time and the results show that the difference between the manual and automatic count is negligible, when grouped in this manner. Being more particular about the languages, such as separating the "others" group into the languages they consist of, proves more difficult, as they are often results of mixed papers, which the language detection algorithm struggles with. Applying the same detection techniques to all papers and counting the total language distribution, shows a relative size proximity of English and Danish papers (Table 2). After cleaning up the data, and removing obviously inaccurate language predictions that are not one of the library options, the distribution is as shown in Figure 6. The issue with this kind of distribution, is that language becomes a deciding factor in a potential list of search results. Non-Danish speakers will rarely be interested in Danish language papers, which becomes a problem, especially in cases with distributions such as that of the study board of psychology, where 90% of the papers are eliminated as candidates from the beginning. There is also a question of whether the different segments of language can even be considered in the same domain semantically, as the words being used are wholly different, except, of course, in those cases where they are not. If a topic is articulated in Danish, are the English results equally relevant? Treating the language separately, would effectively halve the potential candidates of a search, unless the words are exact cognates. It is fair to assume that the frequency of cognates varies between scientific domains and that domains with a lot of acronyms inherently have more. Additionally, between Danish and English, there is presumably an imbalance of how many Danish loanwords make it into the English language, compared to how many English words have been adopted in the Danish language. Therefore, the impact of language diversity, will be different between the study boards, as some domains - particularly in the domain of IT, which is dominant, there are a lot of technical terms acronyms and abbreviations, which transcend language. Figure 6 shows the distribution of language by study boards. There are boards which are monolingual such as the previously discussed study board of international affairs. Also monolingual are study boards of law, cross-cultural studies and philosophy. However most are multilingual with mixes ranging from 90+% of one language to 50-50 Danish and English. The con-

	included papers	mean	std	min	25%	50%	75%	max
Pages	41247	76	48	0	42	71	101	1547
Words	41247	21066	14902	1	10673	18858	28235	266693

Table 3: Averages and min-max pages and words of total papers

sequences of mixing languages however, might not be the biggest issue and lead to inaccurate paper recommendations. The worst case is receiving a recommendation, which is in a language the user does not understand, because there are very few words which mean one thing in Danish and something completely different in English.

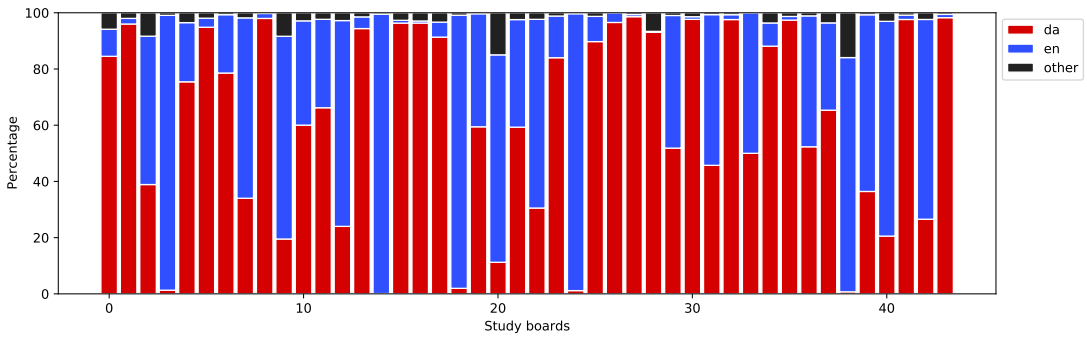

Figure 6: Language distribution of all library papers by study board

Length of papers

Papers in this corpus are of a significant length, compared to traditional scientific papers, which are usually much briefer summations of the work that has been carried out. They contain documentation of all phases of the process in full detail. All the idea generation, research analyses and so on, which are usually shortened extensively for publication purposes, as much of it serves to demonstrate academic competence, rather than the results and methodologies, which are primarily what is on display in traditional published papers.

There is a large disparity in word count among the educations and study boards as seen in Table 3. The median appears to be at 71 pages and 18858 words, however that includes the outliers. Looking at the average number of pages of 76 in Table 3, there is a large distance between the mean value and the maximum value of 1547 pages. In the other end, there are also papers with zero pages and one word, which seems to indicate papers which cannot be interpreted. Seeing as the 75th percentile is 101 pages, it is clear that 1547 and zero are outliers and not to be expected of a significant portion of the corpus, and since more than 50% of the papers are shorter than the median of 71 pages, it is clear that the mean is affected by the large outliers and had outliers been discarded, the mean would drop. Papers such as this outlier also begs the question of how much of the papers' content is actually useful.

Of the 1547 pages of the longest report, approximately 1487 consist of various appendices including data sets and project logs, which leaves 60 pages for the core project work. Appendices are certainly important to the project, but whether they contribute accurately to a paper's "identity" is harder to say. Among the study boards, the word counts range from 11.321-33.169 and the words are evenly distributed among the boards in this range, as shown in Figure 7. The word counts in many cases also reflect the page count, but as discussed previously, there are papers, particularly those from highly creative educations that contain many images, rather than text. It is reasonable to say that the typical paper is approximately 70 pages long, with approximately 20.000 words. Including the large majority, sets the range to about 40 to about 100. That includes all the pages in the papers, not just the ones with useful content, however it gives an estimate on what the size of documents are and thus what the size of the corpus will be.

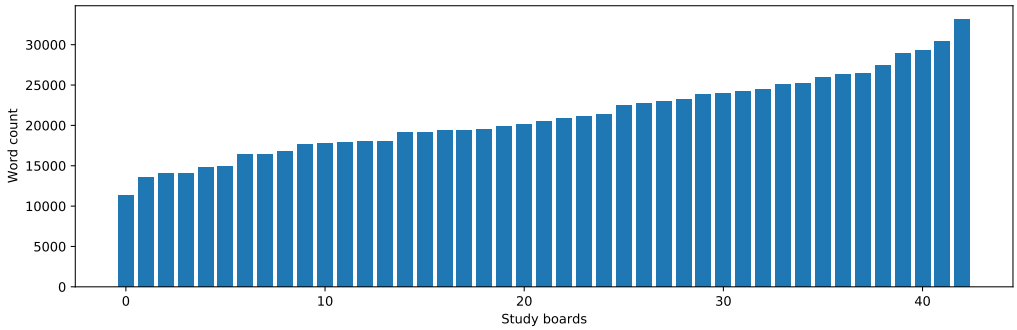

Figure 7: Average word count by study board

2.4 Paper interpretation

During this analysis, some interesting phenomena were discovered. There seems to be a significant amount of papers that have been converted to images, i.e. they have lost their ability to be parsed without optical character recognition, making it much harder to get accurate textual information. In these cases the available information is the meta-data, e.g. title, page numbers etc., and not the actual content. Another discovery is that there seems to be a lot of remnants left over from accidental uploads, where students have uploaded the wrong version, resulting in two nearly identical papers, students uploading just the front page of their paper or students uploading a different document, like the hand-in receipt. Some of these are obvious due to the page counts, but two nearly identical papers, with just a last minute change in meta-data, are hard to find. Since these papers contain content which can be interpreted, the number of papers than can be used for this project is reduced from 54.791 that are available on the library, first to the 41.785 papers which are downloadable and associated with a study board, then to the 41.247 that can be parsed.

2.5 Scope and limitations

This section describes the boundaries for the project in terms of what can be expected, and is largely an evaluation of what is reasonable within the allotted time period. It is quite difficult to arrive at a definition of done, when it comes to information retrieval and recommender systems, because there is no objectively correct solution for recommendations, which is perfect in in all circumstances. There is always something to add, which might have an effect on the performance, but also might not. The actual improvements to a commercial recommender system may not be evident before it has been fully implemented and thoroughly tested, which can take a long time. It is somewhat more likely to reach a definition of done, with regards to the user interface, yet still there has been limited iterations of user interface prototype, because of the time constraints. The system is a stand-alone web application which demonstrates the functionalities of an API for information retrieval. The business logic is separated from the user interface, and is also usable outside the context of the user interface. The data foundation is student papers from the AAU digital project library, but not all are included. In order for a paper to be considered for processing, it must fulfill the following requirements:

- The paper must be associated with a currently active study board
- The paper must be downloadable from AAU digital project library
- The paper must be text and not an image
- The paper must be longer than one page

All papers that do not meet these requirements, will not be used in this project, because it has been decided that they worth the effort of making them useful. Large scale user testing of the recommender system was deemed not feasible, since student reports are not publicly available and the system is only usable through an isolated locally hosted service, meaning users would have to be met in person. As this project ran its course during a world wide pandemic, it was decided that a different approach had to suffice, to ensure the safety of all involved. However it might be possible in the future, following the implementation of an authentication layer, which allows students access from anywhere. The authentication layer was a late addition to the system and thus not in place during user testing. As the user testing was in a controlled environment, it was also decided that the determining factor for a user to be a part of the test, was whether they have written long length project papers in a university context. It was less important what they project was about and which university they had studied at, having done such a project, i.e. a bachelor or master's thesis at most universities, provides enough background for the subjects to put themselves into the situation of having to write another one, or finding interest in others'. Beyond requiring authentication, there are few additional security considerations. These are limited to security by design factors and the system is not thoroughly protected against malicious actors.

2.6 Problem formulation

Based on the problem analysis, which show a lacking search mechanism in the current library, two angles are revealed with regards to the problem of encouraging students to use the contents of the project library. They can be combined into a rather simple statement: ”Demand as little as possible from the user, while still providing accurate suggestions of student papers”. The first challenge lies in making the request for papers as convenient as possible for the user, such that they are not turned off by the effort required and expectations of them. The second challenge is to deliver the desired information to the students, based on the input that was convenient to them.

The main problem formulation:
”How can an information retrieval system be utilized to encourage students’ usage of AAU’s Digitial Project Library?”

Sub-questions:

- ”How is sufficient input conveniently provided?”
- ”What should be retrieved by the system to satisfy a student’s input?”
- ”How should a user interact with the system, to receive its value?

3 Methodology

Project management and software development have a tight connection in this process model that encourages flexibility and change throughout all phases. This section presents both elements' effect on the work produced and the way it unfolded.

3.1 Project

The project management approach for this project has been agile, consisting of iterations of multiple stages during the process. Starting from a problem, the steps towards a solution, were exploratory and analytical - going in many directions. An analysis of the problem revealed the challenges that had to be overcome, however multiple cycles of research, experimentation and reflection were all integral to dealing with the discovered challenges, as well as undiscovered challenges or issues that arose during these cycles.

3.1.1 Literature review

The review of literature has followed the cycles of the project as iterations lead to new areas that required research. As such, each cycle has a preliminary research phase, which involves extensive literature review and also continual research throughout the cycle, to fortify knowledge on the topics that may emerge. The initial research targets reviewed academic papers that are of interest to the problem formulation and challenges faced during at that point in time. These were sourced mainly through the digital libraries of ACM [2] and IEEE [3], where knowledge of recommender systems and NLP was gathered. Other sources were used to complement these, such as various journals and books describing the domains.

3.1.2 User testing

Usability tests are used in this project to investigate the needs of the users in terms of function and user experience. The users of the solution are highly similar, in terms of demographic, in that they are university students. They fall within a similar age range and have completed a secondary education. Their interests and studies are diverse of course, and their approach to the solution will be colored by their backgrounds. However, that is the point of multiple subjects. A group of five university students were selected to perform a think aloud test [4]. According to [5] the optimal number of participants is five. The argument is that for every new participant, the new problems discovered is reduced, because there will be overlaps and repetition of previous participants. Earlier research by Tom Landauer and Jakob Nielsen [6] resulted in a formula for how many problems are found for each new subject. They

describe a Poisson model:

$$Found(i) = N(1 - (1 - L)^i) \tag{1}$$

It considers N total number of problems, proportion L of problems discovered per user and i users. Figure 8 shows the curve with ten problems to be found for a solution, as an example. Using five test subjects, the first round of tests will have found approximately 85% of the problems, based on a 31% average of found problems per user, across the projects they studied [6, p.207]. Obviously, 85% is

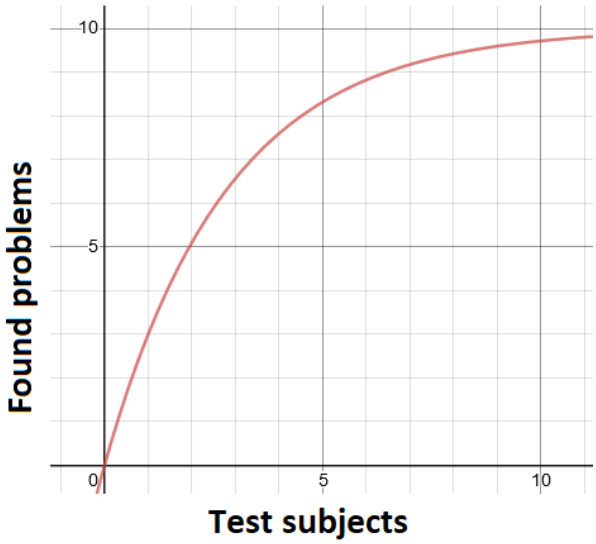

Figure 8: Curve of found problems by number of test subjects

not 100% of the problems, however [5] suggests that doing a redesign and fixing the 85% of issues, to then do another test, is much more cost efficient than getting to 100%, since it could require at least 15 subjects to get there. Instead, after the redesign, a think aloud test with five subjects is run again, to try to discover the last 15% of problems, as well as to discover problems that may have been created with the redesign. Following the curve, this second group of five subjects brings the total found problems near 100%, at approximately 98%. This kind of coverage is very acceptable for a system that is not going into production, but serves as a proof of concept.

Think aloud tests were chosen since they fit well with an agile process [4], as they can be used in all the stages of development. They are not meant to produce any quantitative or exact results, meaning they are still applicable when the prototype to be tested is still in very early stages, and some functionality exist only in the imagination, e.g. a paper prototype, where thoughts and ideas are more useful than correct answers. Further, putting the product in the hands of the would-be users, give direct user declaration of problems and effectively leads to a collaboration with the users.

Tasks and feedback

The five test subjects were initially strictly asked to perform two tasks, and actions beyond those, were

voluntary and exploratory. The first task was to make a search on the website for something that was of their personal interest. This task demonstrates the capabilities of a typical academic paper search engine, and is familiar to the subjects. The familiarity with the function, makes the users able to focus on the form and provide thoughts about the user experience of searching, before moving to the second and more complicated search task, which introduces the subjects to the core function of the solution. At this point they can focus their thoughts on the functionality. The objective of the second task was initially to use the content of a Moodle [7] course page as the input for a search. During the tasks, the subjects were allowed to speak about anything that came up, however occasionally, they were asked to clarify their thoughts or opinions, which may have been unintentionally biasing the test subjects slightly, by making them think some parts were more worthy of clarification than others, and therefore something they too should focus on more. In the same manner, they way in which the clarifying questions were asked could also have unintentionally biased the behaviour, e.g. after a subject declares that something is missing on a page, the replies given by the subject when asked to clarify were different when asked "what" is missing and "where" something is missing.

After each round of testing, new functionality has to be added, and the tasks are adjusted to include these, e.g. the tasks are initially just different types of input, but as filters and other categorization is added, these become part of the tasks. The things that were said can be grouped into a four categories, to create some order in the summary of the interviews, and to more easily compare the thoughts of the subjects to each other during analysis:

- C1: The paths for arriving at the value
- C2: The user experience of arriving at the value
- C3: The perceived quality of the value
- C4: The ways that the value could be applied

Category 1 contains the thoughts related to the first impressions of the prototype. These thoughts are both related to what the system is about and how they should interact with the system to receive whichever value is provided. While they are not necessarily correct, they reveal what the system looks like to actual users. The second category contains the thoughts related to how the user experiences moving around the system before eventually getting to the value. Without necessarily having a specific end-value that is true for each user, these remarks address the subjects' personally selected areas, that affected their experience positively or negatively. Some of the thoughts in this category are linked to thoughts held in other categories, because the nature of user experience, causes it to potentially have an impact in many aspects, e.g. the perceived accuracy of a search engine, can be affected by how a subject had envisioned the results in their minds, rather than what is objectively accurate from the input. Thus, the idea is to separate user experience, from perceived accuracy, to be able to address the

thoughts both individually and combined. This gives the third category ”The perceived quality of the value”, which are simple statements from the subject about whether the results are what was expected under the circumstances, i.e. the input that was given. The interview evaluates more-so the usability of the solution, rather than the accuracy of the prediction, yet they are asked to rate the accuracy. Users are questioned about their subjective opinion on the relevance of the results they are presented with. However, this is to investigate if there are faults or incorrect assumptions with regards to the users' hope for the recommendation that go beyond the technical accuracy, e.g. the results should have been restricted in one way or another. They informally evaluate the top N results as a whole, thus results may be considered ”relevant” if a subjectively good match is presented, even if it is not the first item in the list. While a more elaborate evaluation and comparison of both the objective and perceived accuracy would be useful, it is not detrimental to the problem formulation, but would be very time-consuming and frankly a thesis on its own, based upon the statements of [8]. The fourth and final category is the subjects' thoughts about what the system and the actual value might be to them. The goal of this project is to encourage more use of the digital library content and support different ways subjects may want to use it, is a detrimental factor. While parts of the prototypes change, many things remain the same, thus the subjects will not necessarily have anything new to add pertaining to every category. This is a consequence of performing the tests with the same participants, as they grow familiar with the system. Had the participants been fresh, for each round of testing, they would not have the pre-existing knowledge of what the system is about and could perhaps contribute more in that regard. The experienced participants however, have the benefit of familiarity and the comfort that comes with that, hopefully leads them to deeper thoughts regarding their desired user experience. Not every suggestion by the users is added to the system, but suggestions are evaluated relative to other subjects, to estimate whether this suggestion would be an improvement to others as well. Additionally, in terms of feedback, criticism is valued more than compliments, as for the sake of simplicity, elements are assumed to work unless they are received comments.

3.1.3 System requirements

Through cycles of research, user testing, experimentation and reflection, a list of system requirements was gradually curated. The requirements change rapidly to accommodate feedback from test subjects, before finally arriving at the limit, based on the scope and time limitations set for the project at which point the requirements are set. Each requirement will be formulated with respect to a complete system and will contain both a description and a reason for inclusion. Further, there will be an evaluation of importance for each requirement, estimating to which degree they are necessary for the core function of the system, i.e. a minimum viable product (MVP) and to which degree they are necessary for the prototype that is built during this project. The demands for the prototype will be milder, as its purpose of for proof-of-concept rather than something which is expected to be commercially viable.

To prioritize the requirements, the MoSCoW framework is applied [9].

3.2 Development

The development methodology follows the same overall principles of agile processes. A changing set of requirements, whether transforming or expanding, calls for prototyping and experimentation, to evaluate the best approach going forwards. This includes multiple forms of continual testing, as the requirements lead to new features in the system. Further, continuous integration (CI) and continuous delivery (CD) principles are practiced, to streamline the validation of the new feature implementations [10]. The source code is managed with Git [11] where changes and versions are kept track of during the course of the project, which helps with maintaining an overview of multiple modules that are deployed separately in different locations. The CI strategy is not fully embraced, as there is no service to monitor changes and perform the test automatically, this has to be triggered manually. The same is the case for CD, where the deployment is manually triggered, yet requires no input after the button press. Code availability and distribution is the most important aspect of using Git, because despite it being a single developer project, multiple work stations are used to develop. In practice, from the perspective of the software, it has looked more or less identical to a team, since there has been concurrent development from multiple devices. Therefore, it seems to reason that version control has been as important as if it had been an actual team.

3.2.1 Prototype documentation

Documentation of the prototype based on final list of requirements, is divided into two chapters, Design in section 6 and Implementation in section 7. The design chapter describes the prototype's higher level architecture, services and functionality using diagrams of the UML standard, herein deployment diagrams, sequence diagrams, etc. The implementation section includes the details of implementation choices, with regards to technologies and libraries, for the prototype which has actually been built. Thus, most requirements that are not categorized as ”MUST”, are not described.

3.2.2 Testing and evaluation

Testing and evaluation is done in different ways throughout the project. The system is tested in terms of both accuracy of recommendation and user experience of the user interface. Both are important to the problem formulation as their individual contributions are more or less useless without the other. A recommendation generator that cannot be interacted with, cannot generate recommendations with any merit, i.e. without an input. Similarly, a great user experience on a user interface does not solve the problem without providing the value of the recommendation. The accuracy of the recommendations is tested based on recall and precision compared to a manual search of a selected subset of papers

and the user experience is tested during the think aloud interviews of the user studies. As part of the development the implemented requirements are tested with unit tests and end-to-end tests, which are required to evaluate the success of a feature implementation, as the feature has to both function on its own (unit test) and work within the eco-system (end-to-end test).

4 Research

This section reviews literature and other efforts in the state of the art of the domains that are found to be relevant to the project's problem analyses and research questions. It will cover techniques of text processing with focus on natural language processing (NLP) in the context of IR systems and it will cover the specific application in the IR domain that is recommendation systems, because recommendation systems are closely related to IR systems, yet they do not usually require an input query.

4.1 Natural Language Processing

The primary data source is in the format of natural language, as such implementation of NLP methods should be considered to understand and process it. Generally NLP methods can be applied to anything which relates to human to computer translations of natural language and is used in many areas today. The research of this report is focused on its application in IR systems, such as search-engines and recommendation systems. Practically every digital library with a search mechanism, will employ IR and NLP in some way or another. [12] reviews main characteristics of natural language processing techniques in the IR domain. IR is the general subject of searching for information in documents or searching for documents using natural language queries. IR models select or rank sets of documents based on queries [13, p.1523] . Queries are considered in the same format as the document and receive the same pre-processing treatments, thus the techniques to compare documents are largely the same as CBF in recommender systems, that compare document representations pair-wise, list-wise etc. In IR systems, there are typically a set of modules with functionalities spanning from lexical analysis of documents, to matching user queries to documents, but these modules depends on the purpose of the system, as not all IR systems are used to strictly retrieve matching documents. Some do summarizations of various media or information filtering [13]. The oldest model of IR is the Boolean model, which is based on set theory. A single term query will retrieve the set of documents where the particular term is indexed (i.e. the terms have been identified and tallied), and if another term is added to the query, the set depends whether the systems logical operator is the Boolean AND or the Boolean OR. With the AND operator, the returned set is the intersection of the sets where the two terms are indexed, i.e. the documents where both are present. With the OR operator, the returned set is a union of the sets where either of the two terms are indexed. It appears that AAU Digital Project Library uses the OR operator for their Boolean searches, as the returned sets contain documents where only one of the terms in a multi-term query is present.

4.1.1 Processing unstructured data

Unstructured data such as text extracted from various mediums can be converted to structured documents. Text is converted into something which may represent a document and the way in which this conversion occurs, depends largely on the medium where the text exists and what the text may represent. The transformation requires very deliberate processing to get from unstructured data, to something which can be labeled and considered structured. The documents can be represented with the Vector Space Model (VSM). The simplest is Bag of Words, which for each document creates a vector of words from a corpus, along with a numerical value denoting the frequency of that word in the particular document. This same approach can be expanded with N-grams, which instead of using single words, uses N sequential words, to capture some context. With Bag of Words models, words that are common across a corpus tend to dominate words that are important for a document representation. Term-Frequency Inverse Document Frequency (TF-IDF) is a statistical technique that normalizes these vectors by measuring how important a particular term is to a document from a collection of documents [14, p.22]. High occurrence of a term within a document makes it significant, however, high occurrence of a term across documents makes it insignificant, as it is unlikely to be an important distinguishing term for a document. The other method mentioned by [15], keyphrase extraction, is less used than TF-IDF for document indexing. According to the literature review of [8, p.25], TF-IDF is the most frequently used technique (70% of reviewed papers) for weighing terms. It has two steps: the candidate identification stage and the selection stage. The first finds all possible phrases from a document and the second selects a few as keyphrases [16, p.3]. These phrases can be selected in a supervised or an unsupervised manner. The supervised way includes having humans define keyphrases for documents in a collection of documents and train a model to select phrases from new documents. The unsupervised way is doing lexical analysis to make clusters of phrases and terms that share a noun. These are ranked based on term and noun phrase frequency. The top ones are selected as the keyphrases. Keyphrase extraction systems do not rely on specific training documents, meaning the models are domain independent.

4.1.2 Text pre-processing

Text pre-processing is more or less an obligatory step for NLP, and in particular when it comes to unstructured data. There are some common approaches which are industry standard and applied to every document before doing indexing, term-weighting, etc., which include cleaning, tokenization, normalization and more [17]. Cleaning is done to get rid of noise in the text. The type of noise in the text depends on how it was retrieved and the desired end result is, but generally it serves to generally remove things that are not useful for the text chunk. That could be headers and footers of files, meta-data tags of HTML or XML, and portions of the chunk of text that is decidedly not useful.

This step is in the grey-zone between data collection and pre-processing, as this could be done during the collection, rather than after, since it is usually the first step after collection anyway. With a less noisy data set, tokenization can be applied. Tokenization splits the text into tokens, which are smaller pieces consisting of N words - also known as N-grams, as mentioned previously. The smaller pieces makes it easier to tackle the different words and elements of the texts, and apply individual processing to the pieces that need it. Normalization is an important step to make the terms equal, such that different grammatical tenses of a word are all counted as a singular term, which is the most basic version of that term. Stemming takes care of the affixes, to get the stem of the words and lemmatization takes care of the words, where the tenses cannot be found by removing affixes, e.g. reducing ”better” to ”good”. This is basically done by looking up the words in pre-built dictionaries and if processing speed was not an issue, it could also be used as a replacement to stemming. The last major part of pre-processing is removing ”stop-words”, which are words that do not contribute to the semantic meaning of a text chunk, because they are extremely common to all pieces of texts, e.g. ”of”, ”the”, ”and” etc. Furthermore, there are some minor processing bits, which some might consider part of the cleaning process, however the way this type of cleaning is approached, depends on how much of the original information should be kept, e.g. should line breaks, punctuation and numbers be removed, and should all text be lowercase.

4.1.3 Long documents

The obstacle of long length papers, is not something that is often discussed in the domain of recommendation, but it is something that has been discussed in relation to content-based classification of scientific papers and since both are primarily challenges of NLP, it is assumed that the discoveries in one field are applicable in the other. [18] introduces a framework to handle documents of arbitrary lengths for the purpose of classification. The approach is to split the document into shorter representative chunks and the main contribution the manner in which these chunks are combined and the effectiveness of different combinations. They experiment with different dataset types: Patents, Wireless Patents (subset of patents) and arXiv.org papers. They use multiple modes, one of which is the attention-based model that attempts to [19] discover the significant parts of a paper on a document, sentence and word level to construct the best word vector representation of a document. Using a technique similar to recurrent neural network, they train a model that can reliably select the words and sentences that contribute significantly to the document meaning. The attention model performs best, but it is also noted that only an abstract is required for high performance[20]. Parting the document is an approach also used by [21] that attempts to classify long legal documents. This paper also abstains from using the attention model mentioned by [19], because despite being developed for sizable documents, it is supposedly limited and not suitable for the length of documents that [21] is processing. The size of which is at least 5000 words. Instead they separate the documents into chunks seemingly

regardless of the content of the chunks and observe how it affects the performance of the classification. The number and size of chunks is adjusted for better results, and chunks are weighed according to their significance to the class prediction.

4.2 Recommendation systems

According to a survey of scientific paper recommendations [15], there are multiple methods that might be considered for a recommendation system, most commonly collaborative filtering (CF) or content-based filtering (CBF). [8] did an extensive literature review of "research-paper recommender systems" in 2016, involving more than 200 research articles concerning 120 research-paper recommender systems (96 of which are peer-reviewed) [8, p.3]. They discovered that of the 120 systems, 55% used some form of CBF, whereas only 18% used CF. The remaining used graph-based, item-centric and hybrid recommendations.

CF is built on the idea that people with similar taste, like the same items. Thus, it requires knowledge of researchers' interests and largely ignores the contents of the papers that are recommended. CBF is an approach which assumes that people are interested in items that are similar to another item that they like. For paper recommendation, this means that the system extracts information from the papers and compare these, to see if they are relevant. When a researcher is requesting a recommendation from a CBF system, only that researcher's interest is considered. One such system would encompass the core elements: Item representation, Profile learning and Recommendation generation. Item representation is the consideration of how items are distinguished based on their features, which are categorized as structured or unstructured. Structured data is almost immediately applicable, whereas unstructured data would usually required processing to give it more structure, before being applicable. [15] lists a couple of methods for item representation, such as Term-Frequency Inverse Document Frequency (TF-IDF) and keyphrase extraction. Profile learning is about the researcher that should be recommended to and their individual interest from their history and action. It is typically based on ratings of other papers or publications they make themselves. Recommendation generation based on the attributes of the paper, along with the attributes of the researcher's profile is done by measuring similarity between the researcher's profile and the candidate paper's by calculating the similarity of their document vectors. Based on these similarities, candidates can be selected based on a threshold or ranked list.

[22] proposes an academic research paper recommendation approach that does not rely on a profile of the user, but is based on an input paper's topics and main ideas. It attempts to emulate a traditional search query, but rather than letting the researchers write a query, it considers properties of the input paper to formulate queries. Here, the input paper replaces the typical input of a content-based recommender system, which consists of a user profile built from previous paper viewings and evaluations. The approach looks at four different parts of the input paper: title, abstract, introduc-

tion and related works. These are the parts deemed most indicative of the paper's content and each part is treated specifically with respect to its format and role in the paper. To find similar papers to recommend [22] uses each of the mentioned sections and compare them individually to other papers corresponding sections, as it is assumed all the scientific papers are semi structured and contain these sections. The comparison result is a linear combination of the sections' cosine similarities. [23] is another content-based approach which utilizes TF-IDF and cosine similarity to recommend research papers. The paper's argument for CBF is the nature of the domain, wherein it is assumed users are unwilling to provide explicit feedback for items. In this paper, the input is a user's query which is like a short search query taken from the users and processed. [24] compares stand-alone and hybrid approaches for recommendation and show that while there is something to gain from combining methods, having an effective content CBF approach is practically equivalent in terms of precision. Three content-based methods where compared with the superior method using only content information, while the others also included information about the user's interest.

4.2.1 Ranking

"Learning to rank" is the technical term for techniques that combine data from multiple sources to curate and order a ranked list of candidates, from a list of items. It is heavily used in search engines, where the accuracy scores matter less than the relative ranks among the items. Ranking is also employed in recommendation systems, which differs from when used on its own, where it typically uses a search query input from the users, because they know what they are looking for, where recommendation systems use implicit inputs. Ranking involves careful adjustments and further evaluations, to decide how the different data features should impact the final score, from which candidates are ranked. This adjustment in learning to rank approaches is done by machine learning, if the available data sets allows it, otherwise it is done manually, intertwined with evaluations. The rankings can be based on semantic accuracy, but combined with e.g. popularity or user interests features, it provides results that are relevant to users, beyond matching a search query. Some examples of where it has been applied [25] are Slack [26] Skyscanner [27] and Wayfair [28]. In the case of Slack, they present a view with recent or relevant items (messages or files). Their statistics showed that the relevant items view was on used 17% of the time and performed worse with regards to the number of clicks per search, compared to the recent items view. To improve the relevant items view they apply learning to rank techniques by first creating a traning set with ground truth labels and training a supervised machine learning model to determine which features were most significant. Age of the message was important, which is not surprising considering the preference for the recent items view. A semantic feature, Lucene score [29] is also significant. This feature is the only semantic feature found (besides word count and formatting) and is essentially an enhanced version of the VSM with TD-IDF weights and cosine similarity, where it allows individual weights to an input query by a user and generally

more flexibility. A group of features found significant by the model are related to a users relationship to other users, where if a user's manager is the author of an item, that item is preferred, or similarly, if a user has a lot of messaging activity with the author. The remaining features are related to the user' activity on Slack, where items appearing in channels the user participates in a lot are ranked higher, so is items the user has interacted with, by starring, pinning or reacting otherwise. Slack grew confident that their algorithms now provided relevant results and decided to run the relevant items search along with every recent items search and put the three most relevant results above the most recent results. Their results show significant increase in search sessions, and decrease in searches per session, thus implying users find their desired items faster. Skyscanner helps book flights for people, and try to find the ideal flight using learning to rank. They are not as open about the features they use, however they discuss their approach openly. Just as Slack, they make a training set with labeled data, take the set of features and then perform A/B tests to evaluate the performance of features. The A/B test is comprised of a modified version with learning to rank features and a baseline version based on price. The evaluations were based on search efforts, i.e. similar to Slack, by seeing how many searches are done in a session. Evaluations also considered conversion, i.e. how many flights were booked from A or B versions respectively. They "found evidence of great promise".

4.2.2 Evaluation metrics

To measure the quality of the recommendation system's provided value, state-of-the-art metrics are considered. Of the 96 peer-reviewed recommendation approaches reviewed by [8, p.9] only 75 were evaluated by the authors. These were evaluated using offline evaluations, quantitative user studies, qualitative user studies and online evaluations. The offline evaluations are concerned with the objective accuracy, and use a metric such as recall or mathematical metrics like F1 score or root mean square error. This makes the accuracy quantifiable in an entirely objective way, but they require ground truth labels for the corpus items. Recall is the number of ground truth positive results that were recommended, compared to the total number of ground truth positives. Precision is the number of ground truth positives that were recommended, compared to the total number of recommended items. F1 score is a combination of recall and prediction and the in its basic form is the mean of the two. It can also be adjusted to account for weight of metric, e.g. if recall is thought to be more important for the overall performance, the value of recall can be multiplied by a weight. [30] experiments show the best performance results in terms of recall come from cosine similarity and Pearson Correlation, which means the performance is better in terms of finding as many of the relevant results as possible, rather than precision, which is in terms of getting as few not relevant results as possible (false positives).

Objectivity, however, may not be the way to go according to [31, p. 123] and [32, p.27], which found that there may be large differences in the perceived quality of the value between objective metrics and user-based subjective metrics. Offline evaluations are criticized for this often [8], because they ignore

human factors that contribute a lot to users' impression of the systems, however offline evaluations continue to be the most common method. The second most common method, which does take human factors into account, is user studies. User studies can be either quantitative or qualitative and the recommender-system research area tend to go with the quantitative user studies. In fact, 25 of the 96 peer-reviewed articles used quantitative methods, and only two used qualitative ones. Common for both, is that the questions asked determine, to a high degree, the value of the outcome, and the number of participants is severely important as well. Therefore, it is hard consider either user study approach the best, and [8, p.10] concludes that most user studies have not been large enough to produce statistically signification results. Further, coupled tightly with user studies, is user experience of the recommender-system. [33] explains the impact of user experience on users' satisfaction with recommender systems. Principally, the user experience is affected by the accuracy, however, as mentioned, comparisons [31, p.450] show that better algorithmic accuracy does not necessarily translate to a better user experience. Instead, the focus is on elaborate user studies and develop a user-centric framework containing multiple experiments to measure subjective factors with user feedback, and find other factors that contribute to a satisfactory recommendation e.g. diversity or personalization [33, p.482]. [34] also investigates these human factors that come into play from user experience and the interface that users interact with to get the recommendations. Their findings show that it is important to users why a particular item was recommended, because transparency increases the users' trust in the system, which in turn increases the trust in the recommendations. The methods used to explain to the users why the recommendations are relevant are typically either content reference, i.e. a tag on the item that is relevant to the search, or a social relevance where an item is presented with an indication of other users' opinion of this.

4.2.3 Multilingual recommendation

CF approaches are multilingual by nature, since they do no rely on the natural language of item attributes, but rather discrete values which transcend language [35]. CBF on the other hand is heavily affected my multiple language among items. Natural language similarity depends on similarity in the language. The topics and concepts of a paper are not language dependent, yet if two items contain similar content topic-wise, but the keywords or content is in a different language they will not be considered similar [36]. The issue is familiar to many search engines and a lot of research has gone into the task of multilingual information access, which is the area that covers retrieving information from a collection in any language based on a query. They consider primarily translation based or concept based approaches. The translation based approach translate the document collection, the input query or both into all the expected languages. Doing offline translation of the whole collection saves on online computing time, when a user waits, but requires much more processing - depending on the size of the collection. However, as long as the input query is short, the response time increase

is limited. The translation itself is also a challenge, as even the most advanced machine learning based translation tools get misrepresent the meaning sometimes. The concept based approach also involves translations, but rather than translating into multiple languages, a ”pivot language” is used, which is a third common language which is based on concepts.

4.3 Web scraping

Web scraping is a way to automatically extract data from websites, by requesting the HTML from a web server, and interpreting, filtering and storing information [37]. As opposed to retrieving data from an API provided by a web site, the scrapers use what is usually provided to persons accessing the website through their browser. Tools to scrape the web are typically coded specially for their purpose in one of an assortment of programming languages - the most popular ones being Python, Java and JavaScript. While HTML has some structure to it, the information to retrieve is not organized the same on every web page, thus tools have to be handcrafted and tailored to the particular purpose. The exception is of course, if the whole web page is stored, for later processing. For practically every programming language, there exists a library which helps developers interpret, search and manipulate the HTML Document Object Model (DOM) that they retrieve. For Python, the most commonly used library is BeatifulSoup [38], which scrapes the information out of HTML and XML documents after is has been retrieved from the server. Web pages are not usually built to be scraping friendly. Sometimes a web site relies heavily on dynamically JavaScript generated functionality, which makes extraction near impossible to streamline with simple HTTP commands. In these cases, tools such as Selenium [39], which has web drivers that operate and interact with a browser in the same manner as regular users, are used. To the web site, interactions from users or a tool like selenium are indistinguishable, as long as the selenium scraper is programmed to imitate a user, i.e. does not perform actions faster than humans would be capable of.

4.3.1 PDF extraction

For this project, the majority of scraping on the web, has the end goal of retrieving the student papers, which are available only in PDF format. According to [37] 70% of the information indexed on Google stems from PDF files an therefore, this is also a significantly researched and documented area. There exists tools, similar to the HTML parsers, to parse PDF files. Most are similar in their function, with small feature differences and parsing choices. Therefore, one tool can be suitable for one purpose, but less so for another. Some popular tools to read and write PDFs with Python are PyPDF2 [40], PDFMiner [41] and Tika [42].

4.4 Service architecture

[23] offers their recommendations through a web site, where users can search and receive recommendations. The authors do not describe the underlying architecture of the service and the technicalities of how the recommendations manifest and get presented on a web site. [43] underlines the importance of defining an architecture for a software system, as it plays an important role in moving from requirements to implementation. Architecture models a high-level and general description of the components and interactions in a system by creating an abstraction layer that does not necessarily concern itself with details, but focus on overall structure. [43] further lists a number of aspects of software development, on which architecture has influence, including aspects such as system comprehension, reuse of components, system evolution, conformance to constraints and implementation accuracy with regards to requirements specifications.

In terms of service-based architecture, Service-Oriented Architecture (SOA) and Micro Service Architecture (MSA) are according to literature review the most popular and recommended choices [43] [44]. This is due to a trend of system and service architectures becoming more modular and loosely coupled, contrasting the monolithic architectures where single unit that contain all the software of the system in one interlaced package. The benefits of modularity are evident from the aspects mentioned by [43]. A user's experience with the system might not be affected by a change in architecture, but the change is certainly felt by the project organizers and developers. The modularity makes software easier to read, as it is divided into pieces, rather than having everything occurring in the same place. Similarly, with separate modules, each can be upgraded or repaired individually while largely avoiding repercussions elsewhere, assuming the interfaces are well defined. Both SOA and MSA follow these principles, however with some key differences between them. [45] conducted a literature survey of papers addressing SOA or MSA and compiled a list of distinguishing characteristics for each service-architecture and compare the different aspects.

SOA is a more than a decade old style that is well established in the industry and MSA can be thought of as a variation of that style [44], which is more modern and flexible. Some of that flexibility comes from the granularity of dictated by the frameworks. Effectively, since no service granularity is dictated by SOA, it can have very course-grained services and several functionalities grouped together in a single service. MSA proposes strictly one unique function per service. This could result in many services and to manage these, related functions can be clustered with a designated delegating cluster-head, to reduce necessary interactions to fulfill requests, while preserving independence. Despite this, there can be cases where so many interactions between microservices are required that it becomes ineffective, and this is a case where less-fine granulation is appropriate [45]. Whether it be through a cluster-head or direct services require a communication interface to exchange data. This is obvious for both architectures and here too, MSA is more strict and propose constraints to message formats

and protocols, which is either HTTP-based REST or a publish-subscribe pattern. SOA allows differing formats and in theory support any protocol, since it is left to the implementer to decide and typically handled by an Enterprise Service Bus (ESB).

There are principles for organizing communication within both architectures. Internal methods for this are typically classified as either orchestrated or choreographed. Orchestrated methods rely on a single central coordinator that coordinates all services to fulfill a request [46]. It interacts with all the services and instructs their interactions with each other. Choreographed services make decision on their own without a coordinator and each decides if they need to call another service. It is common to have chains of service calls, among services, using this method. Both approaches are used in SOA implementations, depending on the specific case, however MSAs use mostly the choreographed approach. For external interaction SOAs will have an integration platform (ESB) that mediates between externalities and the internal services and also has the capability to transform the data into a form fitting the internal services' expectations. Instead of an ESB middleware, MSAs have APIs that are consumer facing, but are much simpler and usually only support single protocols and message formats [47]. These APIs are still abstraction layers that make it convenient to utilize the services, but to a smaller degree than SOAs.

In terms of development organization, MSA development teams are centered around the features of the system, and one team is assigned one feature to build a microservice for, including both the implementation and the operation. The combination of implementation and operation refers to the field of DevOps [48], which is an ensemble of techniques and guidelines to integrate software development with the operations of the software. MSA has shown to be highly effective in this regards, as the lightweight and modular nature of the microservices allow agile processes and result often in high efficiency of development resources. This is less so the case with SOA [45], where implementation and operation of a service are handled by multiple teams, which can pose challenges for the process of finalizing and delivering the software, since different teams have different areas of expertise and cross-team communication is required. MSAs thrive on the independence of services, as well as teams because the development processes are easily steered towards the concept of continuous delivery [49], which encourages fast delivery to be able to react to changes and faults before end of the day. Moving software to production environments means going through the entire life-cycle of service, which becomes more time consuming as the service grows. [50] applies the concepts of MSA and observe the DevOps effects, specifically relating to continuous delivery. A large benefit is independent deployment of changes because it eliminates the need to merge a shared code base, thus reducing time spent resolving conflicts in code and removing the time spent deploying code that was not changed. Other benefits mentioned by [50] include simpler deployment procedures and the possibility of zero-downtime. To deal with the increasing number of services, some heuristics are suggested. No complicated interac-

tion between services, constrain service output while being tolerant of input and wariness of backward compatibility when updating interfaces.

4.5 Cloud services

There exists tools and platforms [51] that offer NLP as a service through web interfaces or web services. Users upload their collection of documents or use the web service API to send single documents. Cloud based platform [51] provides a number of services, including language identification, tokenization, chunking,classification and more. However, this is small scale, compared to the solutions provided by Google [52], IBM [53] and Amazon [54] that each offer extensive NLP services to "reveal structure and meaning of text" without any pre-existing experience. They offer these services at a cost, of course, with the enticing offer of getting started for free. [55] reports some extreme numbers for training of state-of-the-art AI models involving NLP that surpass $60.000. Some training costs of other large AI models are Grover-Mega[56] of University of Washington at $25.000 and Google BERT [57] at $6.912. Costs of training are based on hardware rental hours and the complexity of that hardware. As such, smaller models could be much cheaper. What is also typically the case with the NLP as a service solutions, is that they are black boxed and usually pre-trained models. There are also open source tools, such as Natural Language Toolkit (NLTK) [58] that provides a lot of the same services, where the user gets closer to the magic and have more control over what happens to the data, e.g. training the models themselves. Some work is required, but it is free. Aalborg University offers cloud infrastructure to students for free [59], on which students can run their own software on cloud devices, thus skipping on the costs of typical cloud services, as long as blank machines are sufficient, i.e. there are no NLP services (yet). The service is called CLAAUDIA and despite the computing facilities still being under construction they provide easy access to powerful virtual environments that can be adjusted to suit the needs of the computations the students require. The cloud solution is on-premise at Aalborg university, which coincidentally, the Digital Project Library is also. Each student is allowed up to 30 instances, with data storage of a maximum of 10TB running at all times. The basic computation services seem to operate without issue, however the project is still under development and documentation is also in progress.

5 Analysis

To help students find papers, the current digital project library employs an IR system. It is a reasonable approach since there is quite a lot of potential with IR systems, as is evident from very successful search engines in services such as Google, YouTube, Amazon etc., but they have not put much effort into realizing that potential and going beyond a Boolean model that cares only if a term is present or absent. It fails at the two tasks of demanding little input and getting accurate results. This analysis will investigate how these tasks can be solved, using the knowledge of state of the art implementations and research, as well as experiment and user test to find what works in practice. The key areas that have been identified, in which analyses based on research has to be directed, is how to represent the student papers in the system, how to obtain an instigating input from the user, and which shape it should take. The mentioned key areas touch on the aspect of creating a good user experience, however special attention is given to this, through the user testing.

5.1 Data collection

The entire system is based on the data foundation that is student papers, thus these should be available to the system for processing. Rather than downloading each of the more than 40.000 papers manually, this requires automation. There is no API for downloading the student's papers, thus they have to be retrieved through the same channels as they are normally. Here, a web scraping bot can be used, which is programmed to download the papers just as a human user would. A user would go to the project library and select an education from the list (see Figure 2). Each education has a list of student papers listed as seen in figure 9. From here, they would press either the title of the paper to get more information, or the paperclip on the right, to download it. This behaviour can be emulated rather easily with simple HTTP request and HTML interpreting tools such as BeatifulSoup. Doing this retrieval once, means there is a data foundation representing a moment in time. However, papers are added to the project library every semester, and thus this data collection should not be a one and done event. Papers should be retrieved periodically, on a basis regular enough that the papers in the system are mostly up to par with the project library. Most of the projects are uploaded to the project library at the end of the semester, but there are many exceptions, where projects are delayed or shorter term, meaning there could be papers uploaded at any time during the semester too. Therefore, papers in the system could be updated on a monthly basis, or even a weekly basis around the end of the semester. The continuous collection requires solid management of the student papers to avoid duplication and waste of resources downloading papers that are already in the system.

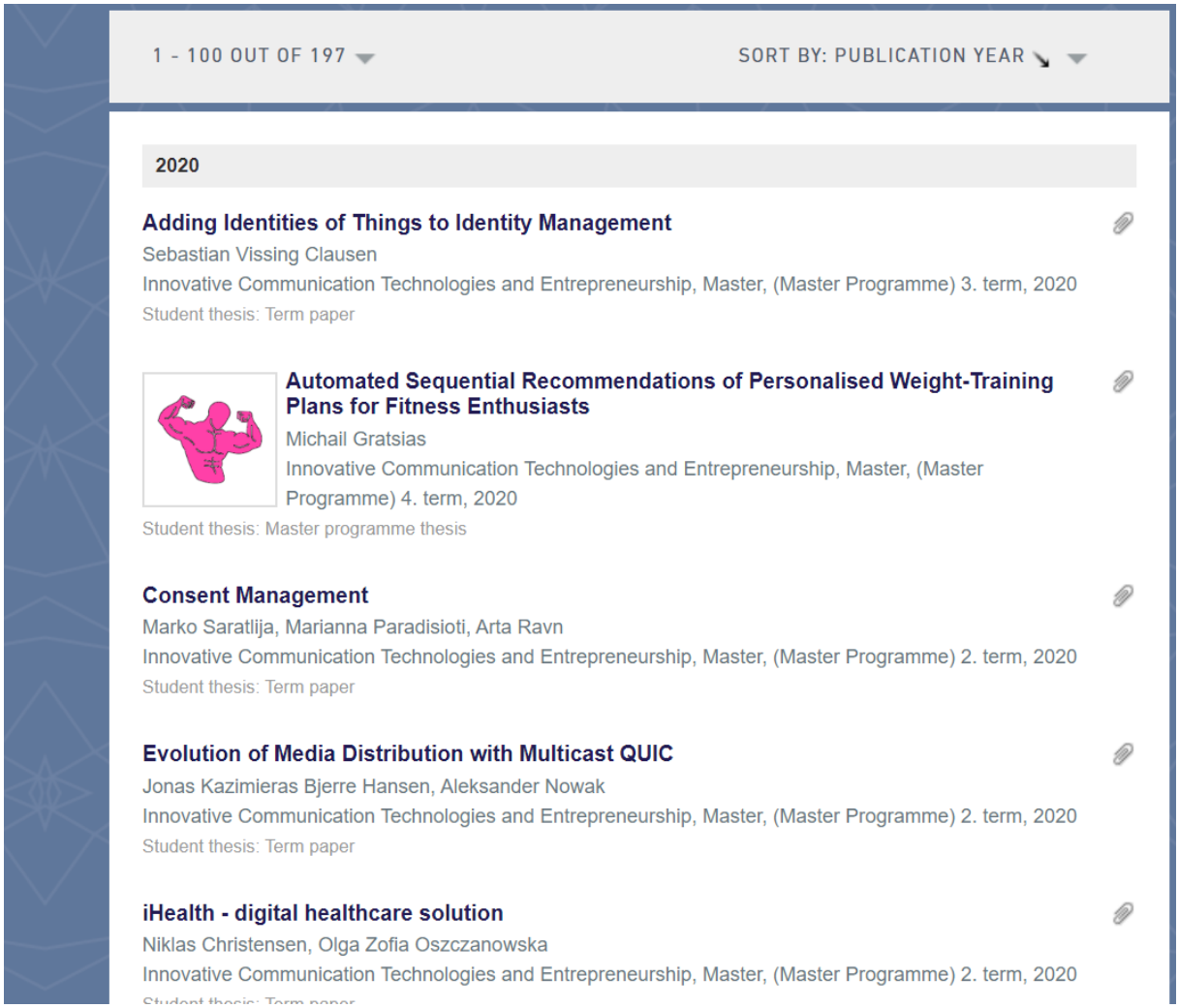

Figure 9: Papers listed for a selected education

5.2 Item representation

There are no user ratings or any history directly associated with the student papers, which makes it rather difficult to use methods such as CF. However, there is plenty of content, making CBF an obvious choice. The first step of CBF is finding attributes of the items, which can be used as distinguishing features. Ideally, all the attributes should be structured, such that there is basis for comparison with other items, so text information which is unstructured (primary content of papers) has to be converted in something with structure that can be adopted into the system.

5.2.1 Selecting item attributes

The attributes that are selected should be available for most, if not all, of the papers, in order for them to facilitate comparison on an attribute level other than has and has not. Even something which is

present on all papers might not be useful, if it cannot be related to an input. Many scientific paper recommendation systems use citation context as attributes for items [8, p.24], however this pertains to paper to paper comparison and requires the input to be a paper with its own citation context. Speculating in having user generated input queries eliminates this option, as they will not have citations for their simple search query. The body text of the papers has this relation, because it is merely natural language. When it comes to document fields such as the the chapters, it dawns quickly that it is impossible to find fields which are present on every paper, apart from the title and meta-data (length, author, etc.). It was discovered during the problem analysis, that there are significant variations in form and fields filled by the student papers. and even comparable chapters can have different labels, which is not as common in the scientific papers being processed in the related works. Therefore, some loss must be accepted, meaning that given a set of fields, any paper without one of those fields, could either not be included in the system or be included, but with less information associated. That means, the fields that are chosen, must be the most descriptive field which is present in most papers. Because it stands to reason that the papers with more information have an advantage when it comes to recommendations, as they have more potential for relevance. However, is that actually the case in the related works? Based on the recommendation systems review by [8], the most used attributes from papers are title, abstract, header, introduction, foreword and references, all with mixed results and the good results have reproducibility issues [8, p.18]. The most consistent results comes from using the densest and most descriptive parts, which turns out to be the title and the abstract. In doing a random sampling of papers, a few chapters are found which are in a very large majority of papers: abstract, introduction, analysis and conclusion. In terms of information denseness, none of the other chapters can compete with abstracts, because it was literally designed with intention of providing readers with an idea of the whole paper's purpose. There is a lot of noise in the other chapters, because of their less descriptive, yet comprehensive formulations. It might be unique to student papers, since they often have length requirements, to write in certain style, where most everything is expanded upon and covered in the paper. Published scientific papers will more often refer to a method or technology rather than explain the whole thing here and now. The abstract appears to be the only instance where the student papers and scientific papers align, in this regard. Using the title and abstract from the papers, makes them more equal, as these are typically closer in length than the entirety of the papers are, which reduces the impact of length as a diversity factor.

5.2.2 Extracting attributes

Extraction of the textual attributes from the papers, based on the above selection, boils down to scraping the abstract and title from the paper PDF files. The title looks to be rather simple, as it also a part of the meta-data, however the title set on the documents does not always correspond to the title of the paper on the project library. Therefore, this title has to be scraped from the web, to ensure accuracy, which

can be done similarly to how the papers were downloaded in the first place. To make things simple, an automatic scraper can augment the paper PDFs with their corresponding titles from the project library, thus making processing at a later time convenient, as the information is already aggregated. Extracting the abstract is more challenging, as this part of the papers come in many shapes and sizes. First of all, the abstract goes by many names. As it is a multilingual corpus, it also has language specific names. The list of names includes, but is not limited to: ”abstract”, ”synopsis”, ”summary”, ”resume” and so on. All of which describing the same chapter. Second of all, the documents are not guaranteed to carry information about sections, as in, where does one start and where does a new one begin. This has to be detected manually by the scraper. There are a few common versions of the abstract format that can be used as general rules. Some papers have a formal title page from AAU, which contains an abstract field (see Figure 10a). Since this page is easy to detect, it is an excellent indicator of where the abstract is found. The issue here is that students seems to struggle with the formatting of this title page and coupled with changing templates it means getting the abstract, and only the abstract out of the page is tricky, because it is difficult to determine where the abstract ends. Another common version is the chapter (see Figure 10b). The abstract is a chapter on its own, with a heading and dedicated page. This way, the abstract is isolated on its own page and when the heading is found, it can be assumed the rest of the page is the abstract. Unfortunately, there are innumerable one-off variations of abstract formats which cannot all be catered to. Inevitably, it leads to document degradation, when an abstract cannot be found, because it an item not having the abstract, means it only has the textual information of its title attribute. After extracting the abstract from each paper it has to be stored somewhere, along with the title and identifying attributes, which makes it possibly to find the paper later.

5.2.3 Vectorizing the item

As a set of attributes has been decided on, they have to be transformed into the vector space. Per the state of the art [8, p.25], the VSM is preferred over alternative representations in most recommendation systems. Similarly for this system, the benefit of making similarity evaluations easily, just by comparing vectors is huge. There are a lot of papers and the approach is still in an exploratory phase, which with another more complex method, e.g. one using machine learning techniques, would obscure some of the nuances of the consequences of the item representation. Instead, with the rather simple VSM, there is a chance to experiment more by observing the behaviour directly. Considering first the most important attribute, the abstract. Even though the abstract/summary part, which constitutes a large majority of the textual content attributes for each items, is actually predominantly English, and even though some otherwise Danish language papers write this part in English, the system still has to consider both Danish and English. There were more languages present in the corpus, but only Danish and English are considered, since they together make up 98% of the corpus (see Table 2). Here is question of whether to keep the languages separate or attempt to do cross-language retrieval. The fact that translation

Abstract

This thesis seeks to explore the social and existential implications of the 'Acting White' (AW) accusation. This is done by utilising the perspective of Tajfel's Social Identity Theory (SIT), Moscovici's Social Representation Theory (SRT), and Sartre's existentialism. There is a specific focus of the notion of conformity throughout the thesis and as the main understanding of the AW accusation is as a strategy for making group members adhere and conform to group norms. The findings of this thesis are that SRT and the concept of schemas from cognitive psychology together with SIT, support an interpretation of AW as a concept that is socially useful for groups. It is used to maintain a group identity by encouraging others to conform to it. It can also serve as a tool to increase self-esteem through group membership. Yet AW also harms individuals accused and can existentially be viewed as self-deceitful behaviour. Lastly, it is concluded that not conforming to an AW accusation should not lead to a greater loss of identity, as the choice of conforming or not conforming in such instances, base themselves on the same underlying strategies.

(a) Dedicated abstract page

(b) Title page with abstract box

Figure 10: Common abstract formats

would only have to be between Danish and English, makes it more likely that this gap could actually be bridged, rather than if there had been twenty languages. In practice, the cross-language retrieval would potentially mean that users searches in one languages received recommendations in the other. In one direction, where a Danish query gives an English recommendation, the user will likely be capable of understanding both the Danish and the English language. In the other direction, where the query is in English, the chance is higher that the user is not capable of reading Danish - meaning those results are not good for anything. There is no issue with papers in multiple languages co-existing, as long as they are dealt with individually in terms of processing, as different languages e.g. require different stop words.

Most approaches start with creating word vectors for each paper, by doing steps of pre-processing on the text and arrive at bags of words. Some of the pre-processing steps are absolutely necessary, such as cleaning, tokenization and normalization. Cleaning is self-explanatory and removes the noise which is not a part of the text or does not contribute to the text. In this case, it is assumed that the abstract attribute already exists and has been separated from the rest of the paper. The text is cleaned, such that unwanted characters and symbols are removed, and so are words that are considered too common. This list of stop-words is extended to fit the circumstances of project writing as there are some commonalities here that appear in many papers. This removal is done for both Danish and English texts. This part is fairly simple, and there is not a lot of choices to be made. Tokenization, on the other hand, is rather complex, because there are different ways to shape and segment the text into tokens, which can change the performance dramatically. The state of the art analyses [30] on tokenization and n-grams in particular, suggests different recommendation performances based on the number of words (N) segmented together in a single token. It also suggests that this performance is based on how similar the documents are to the input document in general. A corpus with documents highly similar to the input, will perform better with high N-grams, because they are likely to use some of the same expressions and phrases. The opposite is true for a corpus highly dissimilar to the input, where lower N-gram (as low as 1-gram) perform better. Thus, the intertextual similarity is what matters and the papers being dealt with in this project have levels of this, due to the hierarchical structure. Generally speaking, it must be assumed that the highest similarity is between papers within an education, second-highest is between papers within a study board and the lowest similarity is between papers across study boards. To cover the lowest denominator of across board similarity, 1-gram, should be tested as a potential size. Since it is a relative measurement, even the highest intertextual similarity found with these papers, is not really that high, because these papers still concern different topics and applications of domain knowledge, but still a higher order N-gram should be tested for documents within the same study boards and educations. After segmenting the words into tokens, they are normalized using the state of the art proposed techniques of stemming and lemmatization. As such the grammatical tenses are removed and all words are put at an equal footing, meaning slight variations of words are counted

as one term. We represent the items in vector spaces by quantifying the normalized tokens. However, since the goal is to best represent what a particular document is about, this bag of words style is not sufficient, as the abstracts are often natural language texts with more than 150 words. Words that are unique for a paper are dominated by those commonly used in general writing. For this reason, terms are weighed using TF-IDF. The weighing is across corpus, however to increase accuracy for domain searches, the IDF part of the weighing for each paper is done across a subset of student papers from the same study board as the particular document. The assumption is that topics being dealt with within study boards overlap. There could be similar topics outside study board e.g. engineering psychology and psychology belong to different study boards, but to emphasize what is truly important in a paper from a study in the psychology board, apart from it being about psychology, the psychology papers must be processed together. This way, unique nuances of each psychology paper are brought to light, rather than the generic characteristics that might represent any given psychology paper within that board. Beyond term frequency, [60, p.112] (reviewed by [8]) also suggests a weighted scoring, where some document fields are granted additional weight. In particular the title, where several propose weighing it up to three times stronger than the rest. As mentioned, the titles of papers are often information dense, without a lot of noise, so it appears safe to boost those terms.

5.3 Recommendation generation

It is assumed initially that more similar documents are more relevant. Therefore the goal will be to compare an input to the indexed corpus and find those which are most similar. From those a number of candidates are chosen, the number of which is chosen mostly arbitrary in state of the art systems. Some inputs will naturally have more similar candidates, thus instead of defining a number, there should be a minimum level of similarity. This is the difference the two standard practices threshold and ranking.

5.3.1 Input

To make recommendations, besides indexing the items in the library, the item that is being used as the input has to be processed as well. Here, that item has to come explicitly from the user, since the system has no previous knowledge of the user's interaction with student reports or their interests. The starting point is important, because to solve the problem of having to know exactly what has to be found, it should not demand much from the user. The ideal would be to get as close to no demands as possible, but infer interest in other ways. Initially it seemed obvious that the link between the student interests and the content of student papers, would be the university courses that students are enrolled in, which due to curricula, are the basis for the content of all the student papers. These courses exists on the open source learning platform Moodle [7], which is the course management system used by

Aalborg University. Seeing as the courses are where the academic topics are taught to students, the assumption is that the corresponding Moodle course page contains content representing what is taught in the course, since this is where teachers make course material available and describe what is going to be taught in the classes throughout the semester. For each semester, the students enroll in the courses they are taking the current semester and it becomes attached to their Moodle profile. This could be useful information which could be extracted and put into the recommendation system, if there was also information about the student's actual interest in these courses. Often times, for university courses, an interest can be temporary while the student is active in the course, after which they focus on something else. Thus, their profile and course history is not necessarily a good indicator of their current interest. When it comes to Moodle, the best bet for when a student is interested in a course, is when they visit the course page. They search out the course, either to prepare for the next lecture or to revisit something taught previously - both times they have a vested and dedicated interest in the topics that are being taught in this course. Thus, the individual courses are inputs reflecting the interest of the student, in the moment. Ideally, capturing that momentary interest would mean having an integration to Moodle that allowed student paper recommendations to be shown right there on the course page. Instead, the experience can be simulated in the recommendation system, which conceptually recreates the moment of interest by receiving the course content and meta-data as an input to the algorithm. As users can not be expected to copy and paste the entirety of the course content into a search bar, the system would have to make it easier for them, by allowing an URL or an ID for the course, from which the service can retrieve the content itself.

There are however some possible issues with using courses as input. They contain multiple topics under a broader title, i.e. the name of the course. This could result in many relevant papers, but none that are exactly relevant, since a paper will usually not include every topic that has been taught in a course, but rather select a few that apply to the current circumstances of a project. It may be confusing to users that were interested in particular parts of the course, since these parts may not be significantly represented in the course input, compared to other topics. This issue is mostly prevalent where course topics have much diversity. More similar topics in a course, means more narrow representation of the course, with higher likelihood that papers written on the basis of the course, include multiple course topics. This difference is apparent from these two course examples: "Identity and Access Management (ICTEC8) (spring 2019)" [61] that is very narrow and "Business Law (ITCOM6) (Spring 2018)" [62] that jumps between unrelated topics. Both are from within the study board of electronics and information technology.

An alternative input could be one that most are familiar with and likely perform on a daily basis, a short string of natural language words, such as when using a traditional search engine. Short inputs, assuming the users has something in mind, are also rather effortless, making it a possible input to base recommendation on as well. A single in-the-moment input is arguably a one-dimensional portrayal of

the student's interest, which does not account for their interest or the context of the interest. The lack of these factors, does however put more of the control in the hands of the student, because there will be no biases based on a user profile and any student will receive the same output from their input, i.e. there is no personalization. There is research supporting the fact that longer queries are more effective [63] [64], it seems to also support the courses as input, as they are usually more than a little longer than typical search queries. It also means that it could be difficult to determine whether courses are actually more useful than short queries, because it might not just depend on the content, but also the length.

5.3.2 Candidate generation

Cosine similarity is by far the most used to determine relevance between documents in VSM implementations. Per [8], eight out of nine of the reviewed systems using VSM used cosine similarity to determine a similarity score. When held against other similarity measures in [30] it performs consistently at a top level, in a range of application areas, including with different levels of intertextual similarity, N-grams and term weighting. [30] also suggests cosine similarity works best when its between the vectors of two documents of different lengths. As the potential for inputs in this system is not limited to an equal size document, there are very good chances that the cosine similarity calculations would be applied to documents unequal in length.

5.3.3 Candidate ranking

The cosine similarity says a lot about the relevance of the papers to an input in terms of textual likeness, but relevance in practice is made up of many other factors as well. A strong factor is recency, based on the state of the art solution by Slack, where this view dominated in the click statistics. Users were most likely to find what they were looking for, in the most recent items. These student papers date all the way back to 2008 and academic research tends to progress quickly, thus there state of the art could be completely different. From a service development perspective, frameworks and DevOps are produced and changed faster than anyone can keep up with and often the old ways of doing this become obsolete. Therefore, the publishing date of the paper could be used to boost the relevance score of a candidate. Usually, with academic papers relevance will be boosted by the number of citation as papers has, however student papers are not cited (hopefully) and they come up short with regards to quality stamps. Grades of papers are not posted along with the paper either. The competence level of a paper is hard to estimate without these, but a possible predictor is the semester during which it was written. Whether it was written on second semester of the bachelor or is a master's thesis, should definitely have an impact on the quality and level of competence on display. Some other information which is unique to student papers compared to regular academic papers, is the motivation for why it was written. Each student paper belongs to an education, which belong to a study board, thus it is

assumed that the papers relate to what is being taught within those groups. The subjects are likely to be similar, but the methods and approaches are even more likely to bear resemblance to others, in particular the ones in the same education group, where the curriculum for the education dictates what the students are taught. If the highest matches are accurate, it is very likely that the user will want to see more of that type of result in the list, thus similar papers should rank higher. These similar papers could also be considered neighbours, such as when using machine learning algorithms to do classification.

5.4 Recommendation evaluation

Of all the evaluation metrics that were researched, offline evaluation is the only one which can reasonably be applied, with decisive results. Quantitative user studies, with explicit or implicit online feedback would require the system to be live in its full version for a while, during data collection. It would also require a large group of students to actively use the service, in order to get sufficient feedback, to conclude anything from it [8, p.4]. It would require a lot of time and would be heavily reliant on other people. If it were to succeed however, it would likely produce interesting evaluation data, that could improve the system, thus it might be reserved for later on, when a complete approach has been decided upon, based on offline evaluation metrics. There is of course no guarantee that the evaluation using offline metrics have any correlation with user studies whatsoever, but if both are conducted they can assist one another. The primary offline metrics measure accuracy based on a ground-truth. The ground-truth says is whether a paper is an accurate recommendation or not. The papers are not labeled by their authors in any way, so no ground-truth exists for the corpus already and thus has to be established first. To establish a set of rules, it has been decided that the criteria for a ground-truth relevant paper, is that the main content, i.e. research, analysis and implementation must have a clear link to the input. Clear link, meaning the paper content must discuss one or multiple aspects of the input as it pertains to the input, and not just as a one-off mention. An example is ”traffic patterns”. Traffic patterns may be discussed in a paper and a course input, but since the paper pertains to road traffic and the course pertains to internet traffic, there is no relation. The final requirement for a relevant papers is that they must be of the ones that were processed successfully into the system, such that it is actually possible to find it. The ground-truth is most often used to evaluate metrics such as precision, recall, f-score, mean square error, etc. Precision and recall has long been the go to duo for IR systems and is still widely used, because they are effective as a top-N recommendation evaluation and because they do not require user ratings. Many recommendation systems has a top-N element in their service which has is has a limited selection of items. This is exactly the case for the system in this project, making precision and recall ideal. Further as a combined measure of those two accuracy metrics, f-score is used, which is an average of the two. There might be an idea to weighing precision higher in the f-score, because in terms of showing it to users, they would likely value the immediate

presence of absence of false positives over the presence of all true positives. Because the number candidates varies, precision results are normalized to account for the finite selection ranges. If there are only five item slots to fill, the precision score must reflect that. Thus, the false positives will be:

$$FP_N = N - TP_N \tag{2}$$

With N being the number of slots to fill, rather than the total number of papers returned. This prevents punishment, when only five of 18 possible are found, because of the limit being set at five. Since the recommender system returns papers with minuscule relevance too, these would all count against the precision score, despite not being expected to be sufficiently relevant.

5.5 Experimentation

As mentioned throughout the analysis, there are uncertainties about multiple aspects of the system that should be tried out. Using the offline metrics as evaluation parameters, these experiments attempt to find the best combination of item representations, inputs and candidate ranges to determine a fundamental strategy for the system. Experiments take place within the domain and across domains.

5.5.1 Datasets and setup

The first experiment considers only the subset of papers that belong to the study board of electronics and information technology. Experimental TF-IDF models are trained for each study board separately and initialised into the system. The study board of electronics and information technology is selected as a subset of the corpus. This study board was chosen because it is the board in which the author of this paper has most knowledge and the best conditions for manually determining the necessary ground-truth relevance of papers. The inputs for the experiments are going to be based on two courses from within the selected study board: ”Internet of Things (ITCOM6) (Spring 2019)” (IoT) and ”Identity and Access Management (ICTEC8) (spring 2019)” (IAM) (see details of inputs in 11.5). The ground-truths for the course topics are as follow:

- IAM has 20 relevant papers across study boards
- IAM has 18 relevant papers in the selected study board
- IoT has 63 relevant papers across study boards
- IoT has 34 relevant papers in the selected study board

Tests are performed on short and long queries of the topics, because the hypothesis is that longer and more descriptive queries perform better, with the caveat of not knowing at which point a query goes from descriptive to noisy. The longer queries serve as a control measure, since course inputs

will be significantly longer than the short queries. If length is a significant factor, it will presumably be apparent in a comparison of long topic query and course input results. The short query will be the general area of the each of the selected courses, such as ”Identity and Access Management” and the corresponding long query will be 200 words from the description section [65] of the topic's Wikipedia entry. Its the same topic, but with more words associated, i.e. a primitive way of doing query expansion and also coincidentally approximately the same length as paper abstracts. The experiments are done with document shapes of 1-gram and a combination of 1- and 2-gram. With the 1- and 2-gram combination, the context recognition is tested and the short input query ”Identity and Access Management” will have a processed vector representation of ”ident”,”access”,”manag”, ”ident access”, ”access manag”. These experiments showcase test using only two different topics, which is not ideal, since the input could be any arbitrary topic and a user would expect the same quality of outcome. In fact, there is an infinite number of potential inputs (if not limited to courses) and extensive testing in this case could be helpful, however two representative cases are selected to avoid manually labeling the whole corpus, to support more topic queries.

5.5.2 Experimenting within a selected study board

Table 4 and Table 5 shows the performance of the IAM and IoT query respectively, within the selected study board. Starting with the topics of input, IAM has a higher accuracy overall. This is not unexpected as the variance of topics within this topic is low compared to the IoT topic. The lower variance, means that even the Wikipedia description and course information did not produce many unrelated results. On the other hand, IoT struggled with this, as the long query and the course input both produced a lot of general security related papers. Security in IoT is highly emphasized in the course, however IoT projects typically focus on other aspects such as service design, data collection, architecture and smart home applications. Thus, there also seems to be a possibility that what is taught in the course on a topic and how it is applied in projects differs. It makes sense, since what students are taught, build upon itself over semesters and not everything from a course is relevant at the same time. In this instance the service design factors and architecture discussion are previously learned tools students applied in the new domain of IoT. The length of the query does not seem to have a large impact based on the performance of the long and short queries, yet the courses were inconsistent. Manual testing with meticulously cleaned versions of the course, showed that the inconsistency is likely due to pollution of the semantic data. There remains too many irrelevant elements on the course page, which obscures the academic topics' representation, such as arbitrary names of people and files that have little to do with the broad academic topics, but concerns the teaching of the class. This is also the case for information about group work, exercises and deadlines for assignments, which is present on most course pages. Names of teachers, teaching assistants and students are all relevant to the students, but not to the topics and in particular the way topics get represented when described in an abstract.

Query	Q length	Method	P5	R5	F5	P10	R10	F10	P20	R20	F20
Identity a...	short	TF-IDF 1-gram	0.8	0.22	0.51	0.6	0.33	0.47	0.5	0.56	0.53
Identity a...	short	TF-IDF 2-gram	0.8	0.22	0.51	0.7	0.39	0.54	0.45	0.5	0.48
Identity a...	long	TF-IDF 1-gram	0.8	0.22	0.51	0.5	0.28	0.39	0.4	0.44	0.42
Identity a...	long	TF-IDF 2-gram	0.8	0.22	0.51	0.6	0.33	0.47	0.45	0.5	0.48
Identity a...	course	TF-IDF 1-gram	1	0.28	0.64	0.6	0.33	0.47	0.35	0.39	0.37
Identity a...	course	TF-IDF 2-gram	0.4	0.11	0.26	0.2	0.11	0.16	0.2	0.22	0.21

Table 4: Performance evaluation of the IAM queries within a selected study board

Query	Q length	Method	P5	R5	F5	P10	R10	F10	P20	R20	F20
Internet of...	short	TF-IDF 1-gram	0.8	0.12	0.46	0.6	0.18	0.39	0.45	0.26	0.36
Internet of...	short	TF-IDF 2-gram	0.8	0.12	0.46	0.6	0.18	0.39	0.35	0.21	0.28
Internet of...	long	TF-IDF 1-gram	0.6	0.09	0.34	0.5	0.15	0.32	0.4	0.24	0.32
Internet of...	long	TF-IDF 2-gram	1	0.15	0.57	0.8	0.24	0.52	0.6	0.35	0.48
Internet of...	course	TF-IDF 1-gram	0	0	0	0	0	0	0.1	0.06	0.08
Internet of...	course	TF-IDF 2-gram	0	0	0	0	0	0	0.1	0.06	0.08

Table 5: Performance evaluation of the IoT queries within a selected study board

Therefore, in the way that the semantic information is applied, the frequent occurrence of names do not provide any useful information, instead their occurrences take the place of other terms that may have been useful. Returning candidates that match a first name of a person is an example of something with a good TF-IDF score, but bad in terms of relevance. Names are not typically used in the paper abstracts, thus rarely contribute anything at all to semantic similarity, when this limited part of the paper is used. There may be other parts of the paper that could have relations to these names, e.g. a title page with supervisors or a bibliography entry. Additionally, on the topic of bibliography, names of teaching material are also typically in the course material and these too seem to not contribute a lot to abstract similarity, presumably because the papers do not mention material or authors in abstracts, but rather in the research chapters and list of reference, where they have space to elaborate. With this type of information, there is more opportunity to find similarity in parts other than the abstract. Perhaps better textual analysis and processing could identify these entities and apply them differently.

In terms of document shape and method, 1-gram outperforms 2-gram on short query and course input, whereas 2-gram is better on long queries. This is based on just two inputs and not conclusive results, but at least logically makes sense as longer passages of text have higher chance of compound words that also exist in the papers. The initial argument for 2-gram, was that intertextual similarity could potentially be high enough within the selected study board and while it did prove effective, the differences are not really that significant compare to the computing time of those vectors. Had time not been an issue, then perhaps both 1-gram and 2-gram options could be a thing, depending on whether the search was in one or all boards.

From the evaluation metrics in the three ranges, it looks as expected, where the highest concen-

Query	Q length	Method	P5	R5	F5	P10	R10	F10	P20	R20	F20
Identity a...	short	TF-IDF 1-gram	0.6	0.15	0.38	0.4	0.2	0.3	0.2	0.2	0.2
Identity a...	long	TF-IDF 1-gram	0.8	0.2	0.5	0.4	0.2	0.3	0.25	0.25	0.25
Identity a...	course	TF-IDF 1-gram	0.8	0.2	0.5	0.4	0.2	0.3	0.35	0.35	0.35

Table 6: Performance evaluation of the IAM queries across all study boards

Query	Q length	Method	P5	R5	F5	P10	R10	F10	P20	R20	F20
Internet of...	short	TF-IDF 1-gram	0.8	0.06	0.43	0.8	0.13	0.46	0.7	0.22	0.46
Internet of...	long	TF-IDF 1-gram	0.4	0.03	0.22	0.3	0.05	0.17	0.25	0.08	0.16
Internet of...	course	TF-IDF 1-gram	0.2	0.02	0.11	0.1	0.02	0.06	0.05	0.02	0.03

Table 7: Performance evaluation of the IoT queries across all study boards

tration of results is found within the first five results and after the first five, it seems to stagnate and relevant results are distributed rather sporadically among the last 15. The precision scores of 0.8 for IAM short and long queries means that four out of five of the results were relevant, which is pretty reasonable for any search engine. The course input for IAM 1-gram had a precision score of 1, meaning that all the results shown were relevant. The results going all the way to 20 candidates are reasonable for at least the long and short queries, where approximately half of the results are relevant.

5.5.3 Experimenting across all study boards

Based on the performance of the subset test and computing time of 2-gram, it was decided to perform tests only on TF-IDF with 1-gram. An observation made, which was not obvious within the selected board, is just how much longer even response time there is, as the query gets longer. Queries of a substantial length, such as those often seen with courses, require a lot of processing time. Every document which contain one of the terms in the document vector is included, when calculating similarity. The input length does not have to be very long, for it to have terms in common, with virtually every document. Table 4 and Table 5 shows the performance of the IAM and IoT query respectively, accros all study boards.

Again, the performance has been generally better for the IAM query. The only time the IoT query is better is the short query. These results show a rapid downward trend in the higher candidate ranges, where it looks as if relevant results are displayed immediately in the top five, and then the list gets crowded with irrelevant papers. The IAM searches struggle with results containing instances of ”identity” and ”management” in social and economic contexts, which are clearly textually relevant, but not the results that were deemed relevant. This goes especially for the short query, where ”management” becomes one third of the cleaned input.

Query	Q length	Method	P5	R5	F5	P10	R10	F10	P20	R20	F20
Identity a...	short	TF-IDF 1-gram	1	0.25	0.62	0.7	0.35	0.52	0.45	0.45	0.45
Identity a...	long	TF-IDF 1-gram	1	0.25	0.62	0.7	0.35	0.52	0.5	0.5	0.5
Identity a...	course	TF-IDF 1-gram	1	0.25	0.62	0.8	0.4	0.6	0.5	0.5	0.5

Table 8: Performance evaluation of the IAM queries across all study boards

Query	Q length	Method	P5	R5	F5	P10	R10	F10	P20	R20	F20
Internet of...	short	TF-IDF 1-gram	1	0.08	0.54	0.9	0.14	0.52	0.7	0.22	0.46
Internet of...	long	TF-IDF 1-gram	0.8	0.06	0.43	0.5	0.08	0.29	0.35	0.11	0.23
Internet of...	course	TF-IDF 1-gram	0	0	0	0	0	0	0.05	0.02	0.03

Table 9: Performance evaluation of ranking application on IoT queries across all study boards

Experimenting with ranking

Since the results were better within a board compared to across boards, it seemed as if there was an opportunity for some simple ranking improvements. As it has been shown that these results can be found in top five, but less so beyond that, a potential way to get more results, is to use the top five to create a new baseline for the results. If the top five are accurate, then more similar to those would likely be accurate too. Based on the analysis of ranking, a few parameters are added to the relevancy score, such as recency, semester and occurrence of study board in top five. The experiments of Table 4 and Table 5 are repeated after applying ranking and shown in Table 8 and Table 9. The IAM queries still fare better, but the ranking showed an improvement for all searches, with the exception of IoT with course input. The main goal was to get the relevant results into the top five and ten of the results, both of which have seen increases in most cases. The performance across the study boards after ranking are at a level similar to within a board.

5.6 Prototyping

This section presents the process of designing a user application facilitating the value delivery to users, based on iterations of user testing and feedback incorporation. The prototype is primarily for design of the immediate user interface, however the demands from users for the user interface, will have consequences for the back-end as in many cases, it dictates data requirements. IR systems of this nature are typically accessible through the web. It does not require much in terms of hardware from the user's end, no utilities such as camera or location sensors. Further, it demands a network connection at all times and does not allow offline usage, which does not leave many benefits to developing the client as a mobile native application, but it is however rather suitable for a web based application. Thus, the protoype will take the form of a web application, from which users can receive recommendations on student papers. The prototyping process will take place as defined in the methodology section 3, and this section akin to a log of the discoveries made during the user testing and how they were

incorporated into the following versions of the system. There are three iterations of the prototype, where the first is based on requirements of the previous sections of problem analysis, research and analysis. The second and third are based on the user provided feedback of the previous version and the feedback of the final version will be collected to complete the iteration and perhaps for future use.

5.6.1 First prototype

The purpose of the first prototype is to have a foundation for the system, which includes both the value creation and the value delivery. The value creation is based on the experimental recommendations and is abstracted for the moment with regards to the prototype, as this part of the analysis will focus on the web application and the users' experience. The requirements for this first prototype include the following simple statements:

- The system must be able to search using keywords or a Moodle course
- The system must provide a way to explore, for users that do not know what to search for
- The system must be able to present a list of candidates from a search

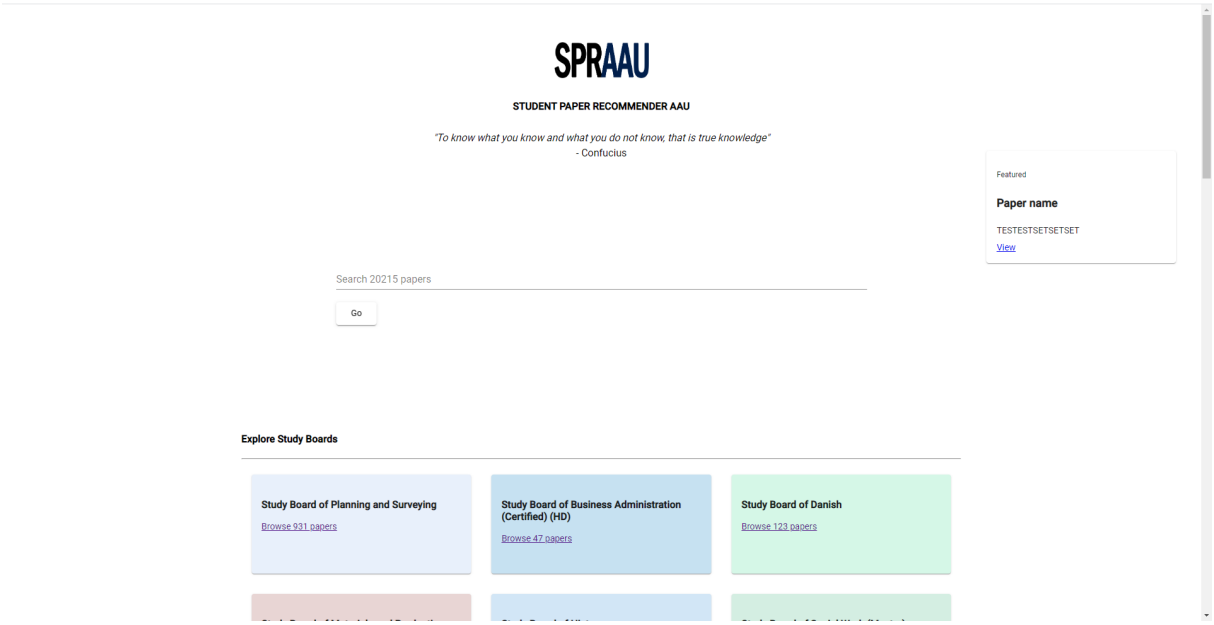

Figure 11: Front page user interface for first prototype.

The front page of the prototype is seen in Figure 11, where users have to option to either search or select a study board, to find papers. The input search is the primary feature and placed in the middle of the page, where an input field informs the user of the library size. The size of the library is helps the user understand the scale at which they are searching and perhaps guides them in their uncertainty. It lets them know how specific they can be in their search. The secondary feature is the cards that allow

the users to browse papers pertaining to specific study boards. Performing a search or clicking a card, takes the user to a second page, on which the resulting papers are shown. If it was a search the results are based on relevance and if a card was clicked, the results are selected randomly. Every time a study board card is clicked, a different selection of papers are chosen. The student papers are presented vertically as cards, containing paper information as seen in Figure 12. Shown on these cards are the

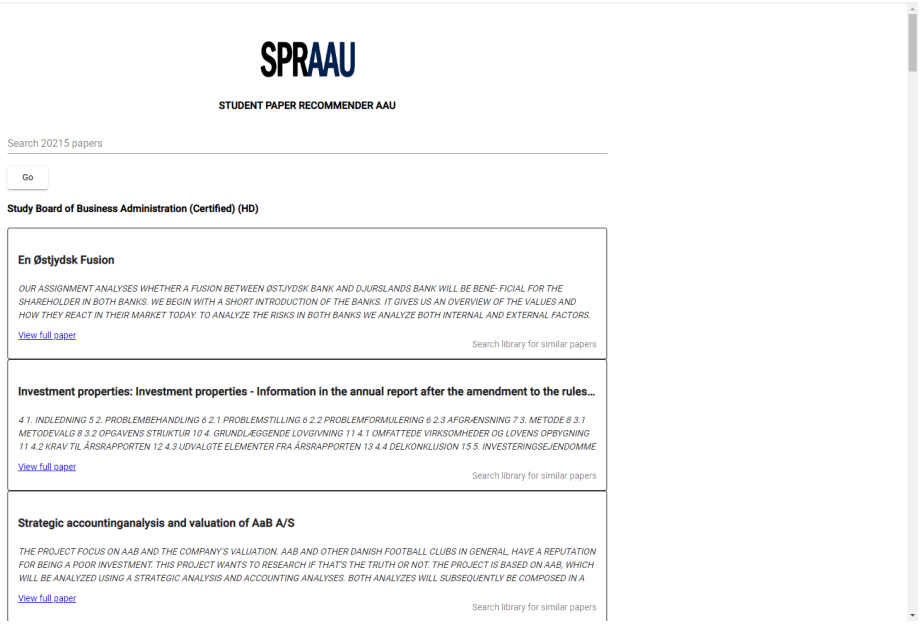

Figure 12: Search results user interface for first prototype.

paper's title, abstract and a link to the full version. The title and abstract should provide information about the paper quickly and if the users are interested, the link will take them directly to a full version of the paper to read more.

5.6.2 Second prototype

The second iteration of the prototype is based on the first prototype, combined with the feedback it received from users. Starting with the front page, several test subjects remarked on the lack of filtering options for a search. Because there are existing categories of papers and many domains, filtering based on study boards or educations is appropriate for this solution, more-so than for a run of the mill scientific paper recommender, where other categorical filtering might be more appropriate, e.g. manual domain category labels and keywords provided by authors. Further, they requested filtering based on year and language. The front page looks as seen in Figure 13, where a filter panel has been added. The panel expands to show the filters, as seen in Figure 15. Where users can choose which study boards and languages they want to include in the search. Thus, each paper must also have information about the language associated to it, which requires manual detection. The search bar was also moved to the top of the page and made to automatically adjust in size with the input, to indicate that longer queries

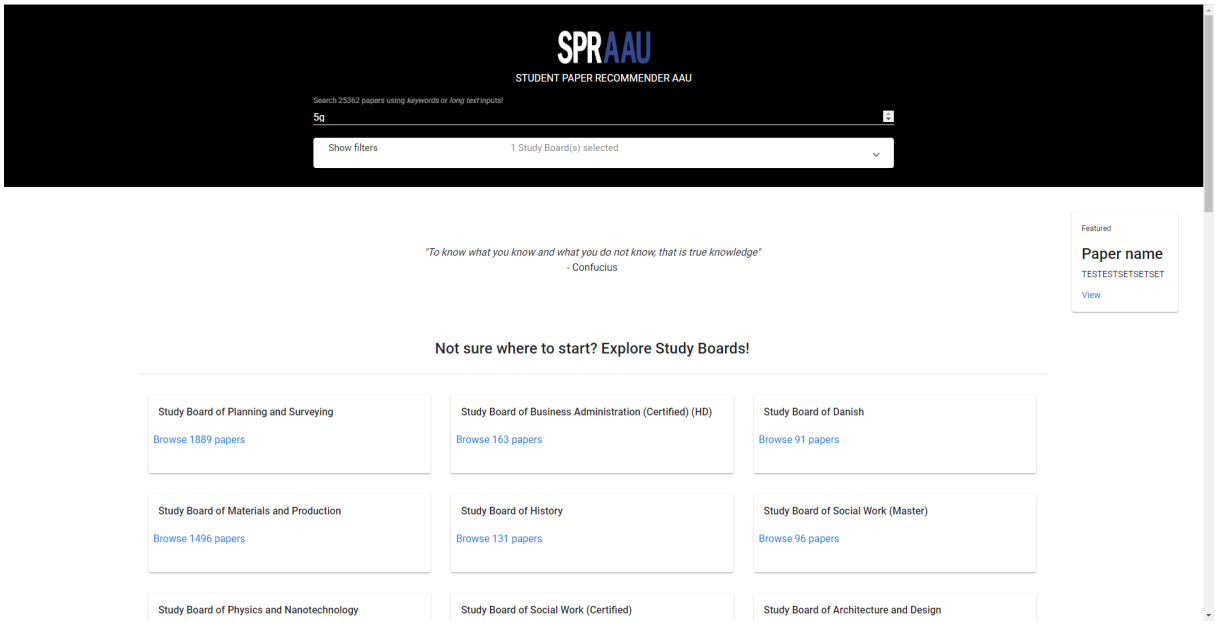

Figure 13: Front page user interface for second prototype.

are possible, which is something users are not used to with search engines. Many users had issue with the way the abstract was presented on the cards, as they deemed it difficult to read and confusing to not have the full abstract available. In this version, the full abstract is added to the card, however it is expandable, meaning the user decides themselves if they want it shown. The cards are shown in Figure 14, where another attribute of the paper has also been added to the card. Since there are no grades on the student papers, users suggested length of paper to generalize to comprehensiveness of a paper and stated that length was important in determining whether they wanted to even read a paper. In terms on the number of results, some users preferred to have as many options as possible, so they could filter them out afterwards. The candidate generation had previously been limited to a selection of those that were most relevant, based on the experiments that showed the relevance degraded drastically. Thus, more candidates are shown and pagination had to be introduced, as well as an indicator of how many results were found, as this number indicates the degree of success of the search. In terms of navigation multiple users mentioned uncertainty after of a search, when nothing happened for a while and they did not know if the page was supposed to change. Since computing the results may take a few seconds, a loading indicator has been added while searching. Navigation was also improved by moving the search parameters to the URL, rather than being internal values. This make it possible to revisit a search, as well as go back to the previous search page that was just left. It is now also possible to return to the front page, by pressing the logo in the top.

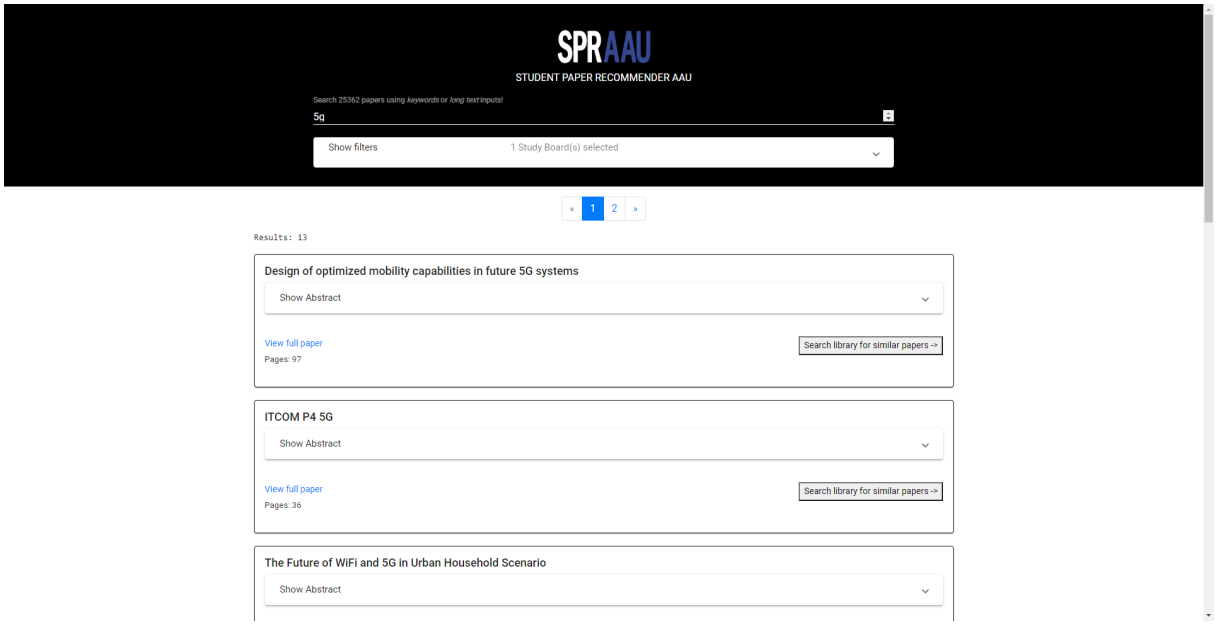

Figure 14: Search results user interface for second prototype.

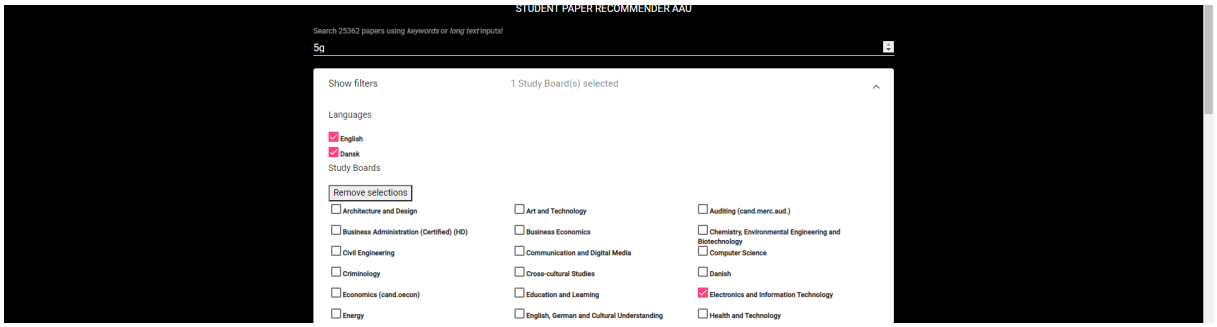

Figure 15: Filter panel for second prototype.

5.6.3 Third prototype

The final and third prototype implementation is again based on the previous version. On the front page, the search bar was widely disliked. Besides looking too much like a part of the background, it also did not have a dedicated button. When just searching, this was fine, because the enter-button works as long as the field is selected, but when filters were added afterwards, selecting the input field to press enter and search was a hassle. Therefore, the search bar was made to stand out more and look like a search bar, as well as being granted a button dedicated to executing a search (See Figure 16). The filter panel, which prompted the search button demand, also received its own executory button, which applies the currently set filters to a new search. The search results page and pagination was largely satisfactory, however feedback from both the first and second version had implied that more information could be added to the card, to get a better sense of what the paper is about as well as estimate its quality without reading it. Here, they suggest publishing year and the semester during

which the paper was written. This information is available on the Digital Project Library, thus this has to be downloaded and associated to the papers as well. The report download service could be updated, to also retrieve this information when downloading the paper. However, downloading the papers all over should not be necessary and instead a service to update meta data on reports is created. Some meta data functionality already existed in the paper downloading service (specifically titles), however a separate service for this may be more appropriate. They also suggested adding topic keywords to the card, based on the content of the papers. The final paper attributes on the cards are shown in Figure 17. While not an attribute, a helpful feature was added to the card which highlights the part of the

Figure 16: Search interface for third prototype.

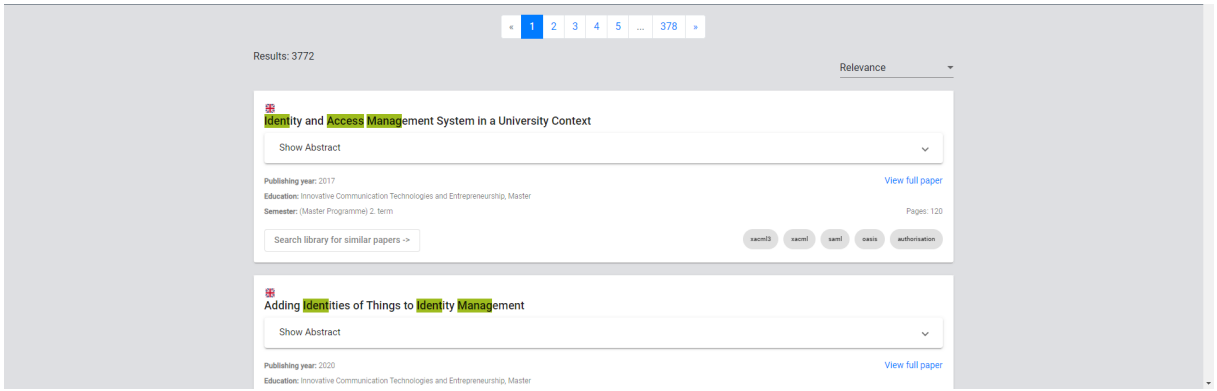

Figure 17: Search results user interface for third prototype.

content which matches the search. Users had a hard to seeing why some of the papers where relevant - in particular if the abstracts were long. The last significant addition is a sorting option, which allows users to sort based on some of the paper attributes, i.e. publishing year and semester.

5.6.4 Final feedback

The feedback on the final prototype will not be incorporated during this project, however it is still relevant for the iteration and the future. Thankfully, there were few remarks made by the users on the last prototype. In terms of the searching, a user wanted the ability to direct their search more and designate words of the query which absolutely had to be in the result. The same user also wanted to make compound word searches, where the results match the word combination rather than the

words individually. A interface related search request is to move the magnifying glass into the input field rather than next to it, since it is more in line with typical usage of the magnifying glass in other searching solutions. Lastly, regarding the candidate cards' layout, a user requested smaller cards, to allow more in the view at a time, by having both rows and columns.

5.7 Service architecture

For a system in a field such as IR and recommendation, where there exist so many different techniques and opinions on what is the correct or best way to do things, flexibility is a must. A lot of techniques have to be applied and tested before knowing if they are worth anything, which requires being able to exchange one functionality for a better version or another thing all-together. This has to be convenient to do, to test as many techniques as possible during limited time, without interfering with the rest of the system, in a destructive way. The popular MSA focuses a lot on decoupling of services and is very helpful especially during experimental phases and initial development, where functionality is added gradually. In this project, a number of functionality demands have already been determined and they happen to be isolated in nature, e.g. the ability to download papers and the ability to process papers are two functionalities that despite having dealing with the same type of data, does not have to be entangled with each other. Papers are downloaded, full stop. Papers are processed. Changing the way papers are downloaded, should not change how they are processed. Thus works as long as the interfaces are standardized and the input and output format of the functions are known. With MSA, the services are smaller and focused around one task, e.g. downloading of papers or ranking of candidates. This type of separation provides the all important flexibility and exchangeability, which can allow a system to evolve indefinitely, by updating services and adding new services, without having to merge a code base or change core elements of the system. Whilst services are independent they still communicate between one another and following practices of MSA, services participating in the system should be choreographed, such that each makes its own decisions and the intermediary services enforce this independence by calling upon them indiscriminately, and allowing the service itself to decide the next step. If the service currently disabled, it will choose to do nothing.

5.7.1 Cloud infrastructure

It was touched upon in the research phase and became evident during experimentation that NLP requires a lot of computing power and memory, when it gets even moderately advanced. Running these services on a regular commercial cloud was hardly an option when looking at the potential costs. Fortunately, CLAAUDIA emerged, offering equivalent expensive server time and computing power for free. As if "free" was not hard enough to compete with already, CLAAUDIA also has the added benefit of running within the internal network of AAU, same as the Digital Project Library, meaning

the otherwise time consuming process of downloading the student papers becomes much faster. Using AAU on premise servers as the cloud is a risk in terms of availability, as it is not true cloud and there are no replica servers on around the world. If the AAU servers go down, so does CLAAUDIA. Thankfully, this only happened once during the project.

5.7.2 Service overview

Based on the analysis, a number of features have been identified which must be supported by the system to create and serve the desired value to users. The following is a short overview of the A simple version of the envisioned complete system is shown in Figure 18. Portrayed is a system running on CLAAUDIA servers, with a web application reaching through to the user. In the cloud, there is a number of offline services, that has processed the papers and a number of front end services ready to handle requests.

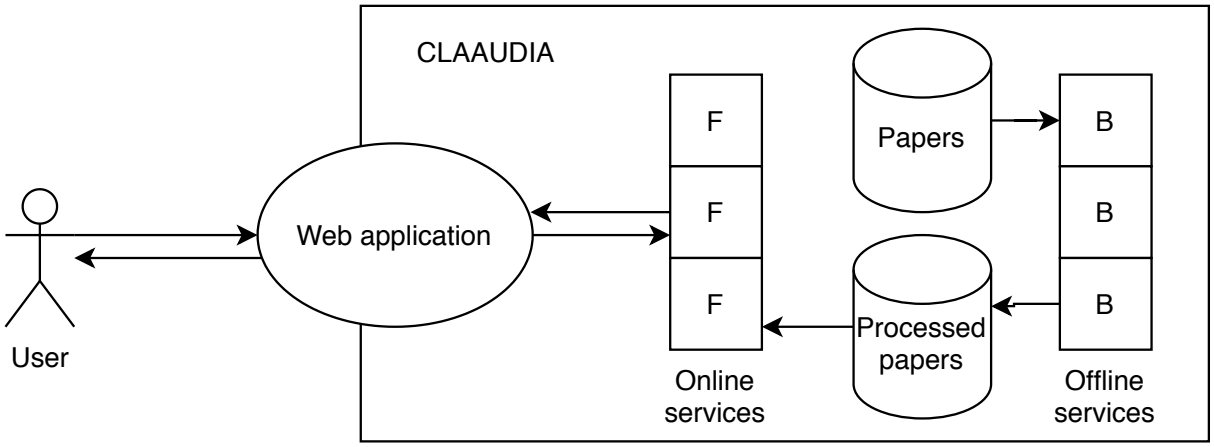

Figure 18: Context diagram of envisioned system

Front-end

From the serving perspective, the first thing that is needed, is the interface, with which users can interact and receive the benefits of the system in a way they can interpret. While such an application can be considered as a single module, this too can be build as smaller modules, which is preferable to maintain the same architectural patterns throughout development. A great platform that emphasizes the modular approach is Angular [66], which is a Google produced framework for web applications that work across all deployment targets, thus ensuring compatibility with all devices that are web-enabled, with a single code base.

The web application will be considered as a single service module in the architectural design going forward for simplicity, which is responsible for presenting data to the users and receiving input from the users. It will also be referred to as the front-end of the system as opposed to the remaining modules,

which are all part of the back-end.

Online back-end

On the back-end, there should be an entry point module and an intermediary module that can facilitate communication amongst the rest. The entry point or gateway receives HTTP requests from the front-end application. This should be a publicly accessible API and the only service other than the interface application which can receive requests from the outside. With a single entry point, the back-end is better protected as all communication has to go through here, and security efforts can be conveniently focused early in the processing. This can handle the HTTPS communication and potential authentication layers, which are not necessary internally. The rest is logically isolated from the outside and they know which inputs they can receive through their interface, because it is funneled through the filter of the API, which can do some early evaluation of the request and discard invalid ones. By preventing direct outside access to the services, their usage becomes limited to that which is dictated by the stringent requests from the user application. It is not possible to tailor requests to individual services that are misuses or unexpected uses of those services, which were not planned for. This is a good thing in terms of security and protection of assets, but not necessarily the best way to provide the most value. As mentioned previously, a recommendation system such as this, could have a place integrated in a Moodle-like platform, but it would obviously have to be adapted to the platform. With openly available services, this integration is much easier to make, as developers can pick and choose for themselves which modules they would like to include. This type of service is more in line with what is offered by the cloud service providers mentioned in the research chapter, wherein they offer some pre-built modules and pre-trained models, but that is obviously a different value than the one that is sought after in this project. It does however, demonstrate an added benefit to the users having control of which modules are included, a concept which could also be added to the recommendation system, since MSA allows it. Moodle is however still involved in the online back-end, as a user can provide a link to a course, which the back-end must extract the content of. When the internal communication facilitator service detects this, there should be a service it can forward the request to, to handle it.

Offline back-end

The offline back-end services prepare the data foundation, such that the online-back end has something to compare inputs to and present to the users. The prep-work includes the download, processing and augmentation of papers and these are not activated by user requests, but rather by trigger events either based on time or changes to data. These services do not communicate with each other or with the outside for their function. All their communication is with local storage as either file reads or file writes, except when they have to be activated.

5.8 Access control

Not every student paper is publicly available, but only available to persons with AAU access. The system must continue to enforce this access control, by requiring AAU login privileges to use the service. The data to be protected is on the back-end and thus authenticating the users for any requests here is essential. As there is a single entry point from the outside to the back-end, this is where the user must be authenticated and validated before use. However, at the same time, it has to work seamlessly with the user interface, such that the front-end and back-end can share the same login session. AAU has several authentication mechanism implemented and two that could be of interest to this project, Where Are You From (WAYF) [67] and Central Authentication Service (CAS) [68]. WAYF is a Danish produced identity federation service used in education and research for single sign-on. When an organisation has WAYF implemented they can sign up services, which authenticate through the organization's WAYF and share information about the user with the service. It is up to the service, make the decision whether to allow access to a user based on the information it receives from WAYF. WAYF has reasonable documentation, however it depends on the implementation of WAYF in the organisation, AAU in this case, as they have to explicitly register the service which needs to authenticate through it. CAS is another single sign-on protocol, which can be used by third parties to authenticate users. It does not require any sign-up and can be used by anyone. The implementation includes an endpoint that the service can redirect users to, and prompt them to login. Following login, they are directed back with a Ticket Granting Ticket, which the service validates with another endpoint of the CAS implementation. Successful validations is rewarded with a Service Ticket, which the user's browser saves and passes along with following requests to the service, to persists the authentication. The outcome of either of these methods is acceptable. The service has no need for meticulous user information, it just needs to know whether they have an AAU login or not. The easiest solution seems to be the one, which does not require the organisations explicit involvement, CAS.

5.9 Requirements specification

Requirements for the prototype are based on feedback from testers during the several iterations, as well as on the multiple analyses of the problem and methods. The functional requirements describe what the system shall do. For the sake of overview, the requirements are described service by service. Each of the identified requirements are assigned to a service which is responsible for that functionality and potentially others like it. All the requirements are formulated as "The service must...", except for the web application, which is considered a system of smaller services on its own. When the requirement description refers to a service, it is only the particular service for which the requirement is written. The requirements for the web application are the only ones that relate directly to the user and how they are able to interact with the system, whereas the remaining requirements pertain to a hidden back-end.

	Functional requirements for web application
FR#1	Recommendation prompt with a course input
Description	The system application must accept an URL to a Moodle course as new recommendation input
Reason	Basing a recommendation on a course is a key factor in reducing the effort required to find relevant items. It is something which the user is already familiar with and is used as a representation of their interest. See section 5.3.1.
Prioritization	**Prototype**: MUST | **MVP**: MUST
FR#2	Recommendation prompt with a text input
Description	The system must accept a string input as new recommendation input
Reason	The user should have the option to get a recommendation based on an input of their own choice, if they have something particular in mind, which is not adequately covered in a course. See section 5.3.1.
Prioritization	**Prototype**: MUST | **MVP**: MUST
FR#3	Recommendation prompt with a paper input
Description	The system must accept previous recommendation results as new recommendation input
Reason	If the student finds an item among the results which is a better representation of their interest than their previous input, they must be able to use that item as a basis for a new search that will yield results closer to what they are interested in. See section 5.6.
Prioritization	**Prototype**: SHOULD | **MVP**: MUST
FR#4	Apply filter to recommendation prompt
Description	The system must have options to preemptively filter recommendation results by study board, language and year.
Reason	The user must have control of the recommendation. The system can go a long way with recommendations, but with more information from the user, the recommendation will inevitably come closer to the user's interest. See section 5.6.
Prioritization	**Prototype**: MUST | **MVP**: MUST
FR#5	Show recommendation results
Description	The system must present the recommendation results with pertaining information, to the user
Reason	The recommendations that are generated for the purpose of showing them to the user that prompted their generation.
Prioritization	**Prototype**: MUST | **MVP**: MUST
FR#6	Sort recommendation results
Description	The system must allow users to sort the recommendation results based on relevance, publishing year or semester.
Reason	Despite the efforts to rank, the user knows best what they emphasize on. Further, this emphasis may change during search and giving the opportunity to sort based on their preference provides a better user experience. See section 5.6.
Prioritization	**Prototype**: MUST | **MVP**: MUST
FR#7	Browse random study board papers
Description	The system must provide a way for users to retrieve random papers.
Reason	The user does not always have an exact entry point to recommendations. Therefore generating a starting point for them, will help them get started, when they do not know what to put into the search bar, but still are interested in an area. See section 5.6.
Prioritization	**Prototype**: COULD | **MVP**: SHOULD

	Functional requirements for API service
FR#8	Publicly accessible
Description	The service must provide an API, which can be accessed from outside the system.
Reason	In order to protect sensitive data, the majority of the system i logically isolated from the outside world. The API service is the only way in. See section 5.8.
Prioritization	**Prototype**: MUST | **MVP**: MUST
FR#9	User authentication
Description	The service must handle authentication of users
Reason	There should be no unauthenticated users making requests beyond this point in the system. As such, this service will have to redirect the users to where they can get authenticated. See section 5.8.
Prioritization	**Prototype**: COULD | **MVP**: MUST

	Functional requirements for Moodle service
FR#10	Moodle authentication
Description	The service must be able to authenticate to Moodle
Reason	When a Moodle course identifier is provided by a user, the system has to be authenticated to access the page on Moodle. See section 5.3.1.
Prioritization	**Prototype**: MUST | **MVP**: MUST
FR#11	Moodle course enrollment
Description	The service must be able to enroll to Moodle courses
Reason	When a course is accessed by the service on Moodle, it must be enrolled into the course, to view the course content.
Prioritization	**Prototype**: SHOULD | **MVP**: MUST
FR#12	Course content retrieval
Description	The service must be able to parse content of a Moodle course page
Reason	The course is retrieved as a web page and from this page, the service has to extract the content that is going to be used as input.
Prioritization	**Prototype**: MUST | **MVP**: MUST

	Functional requirements for Candidate service
FR#13	Relevant candidate generation
Description	The service must be able to compile a list of candidates based on vector similarity
Reason	Cosine similarity of TF-IDF vectors finds the papers most similar to the input. This is an important step in making the recommendation, as these are the papers that will eventually end up as recommendations to the users. See section 5.3.2.
Prioritization	**Prototype**: MUST | **MVP**: MUST
FR#14	Random candidate generation
Description	The service must be able to compile a list of random candidates from a study board
Reason	This service has the information of candidates readily availbable and thus can be expected to provide the random generation service, which is asked for by the web application. See section 5.6.
Prioritization	**Prototype**: MUST | **MVP**: MUST

	Functional requirements for Ranking service
FR#15	Ranking based on defined factors
Description	The service must rank selected candidates based on decided parameters.
Reason	Tweaking of the ranking is done to improve the perceived accuracy of the recommendations. The combinations of parameters will add to the relevance score and give the user results that are likely to seem more relevant. See section 5.3.3.
Prioritization	**Prototype**: MUST | **MVP**: MUST

	Functional requirements for Downloading service
FR#16	Download papers
Description	The service must download student papers from the digital project library that are not in the system already
Reason	The data source for processing is student papers and more papers makes it generally more likely to find accurate recommendations. See section 5.
Prioritization	**Prototype**: MUST | **MVP**: MUST
FR#17	Download paper meta-data
Description	The service must be to download meta-data for each paper from the project library
Reason	Not all information which was requested by test users can be extracted from the paper itself, but it can be found on the digital project library. See section 5.6.
Prioritization	**Prototype**: MUST | **MVP**: MUST

	Functional requirements for Language service
FR#18	Detect language of papers
Description	The service must be able to detect and annotate the language of all papers
Reason	A common request by test subjects for a filtering parameter was language. The language of a paper was also a large diversity factor, which was necessary to manage, for analytical purposes. See section 2.2 and section 5.6.
Prioritization	**Prototype**: MUST | **MVP**: MUST

	Functional requirements for Topics service
FR#19	Generate topics for papers
Description	The service must be able to generate keywords for the topics of a paper
Reason	Test subjects requested keywords directly, but also stated that the abstract was too long, when they just wanted a quick idea of the paper's content. See section 5.6.
Prioritization	**Prototype**: MUST | **MVP**: MUST

	Functional requirements for Processing service
FR#20	Processing papers
Description	The service must process and prepare papers for candidate generation.
Reason	The papers need to be indexed with TF-IDF weights, to gain a structure, which can be used for vector comparison. See section 5.2.
Prioritization	**Prototype**: MUST | **MVP**: MUST

6 Design

The system design is based on the experimentation and prototyping during the analysis, and use the requirement specification as a blueprint. These designs are meant to combine the information from previous chapters to define the system in terms of architecture, processes and functions, so that it is implementable. It covers both elements of high-level design and more specific design choices. Focus is on the back-end services, and less so the user application, as design choices have already been presented in terms of looks and form, during user testing, and will only be described in terms of the system's high-level architecture. The back-end will be looked at in terms of its arrangement, internal and external relationships, and management.

6.1 Architecture

Using MSA principles the system is architectured to spread functionality to multiple services, rather than a monolithic entity. As a part of the requirement specification, each requirement was assigned to a services, such that functionality was distributed between a number of smaller entities. All of

these service must be independently deployable on a server, meaning each must be a full stack micro application, which has the capability of self management and autonomy.

The architecture of the system is best illustrated with a deployment diagram (Figure 19). The diagram shows two CLAAUDIA server instances, such that the web application is separate from the computational parts. The back-end is not dependent on the user application, and can be used as a stand alone API service, thus it makes sense to separate the two, to minimize the potential points of failure for the API. Communication between a CLAAUDIA instance and the outside uses HTTP, and so does all communication among the internal online services. The simple synchronous messaging protocol is recommended for MSA, because it keeps the pipes dumb and leaves logic to the endpoints. Per the deployment diagram, the system involves two external services as well, both of which are hosted by AAU. CAS is the SSO provider necessary to take this system public, as it can restrict access user with an AAU login. There are various opinions on how MSA should be organized with regards to interservice communication, where some suggest chaining is the absolute worst and others saying chaining is actually encouraged, as long as asynchronicity is practiced. The online part of the solution is not ideal in that regard, because it does include some chaining, which is not asynchronous, because the services use inputs directly from other services.

6.2 Services

The following describes the sequence of the services' primary actions, with regards to their extra-service communications. All the services have been identified for the prototype, and they have been definitively grouped into the category of offline services and online services. Offline services prepare beforehand and online services handle requests as they arrive.

6.2.1 Offline back-end services

The curation of the dataset is visualized in Figure 20, which shows the offline services' interactions. The topics, processing and language services are started first, and start waiting for papers to be downloaded at which point the services start to perform their actions in parallel to create an augmented dataset from the papers. After processing, they go back to listening for papers, ready to repeat the task when a new paper arrives. All the services are autonomous, where the downloading service is based on a timed event trigger, whereas the remaining services are based on a file creation event. Such arrangement follows the MSA pattern well and provides the benefits of almost complete independence and exchangeability of the components.

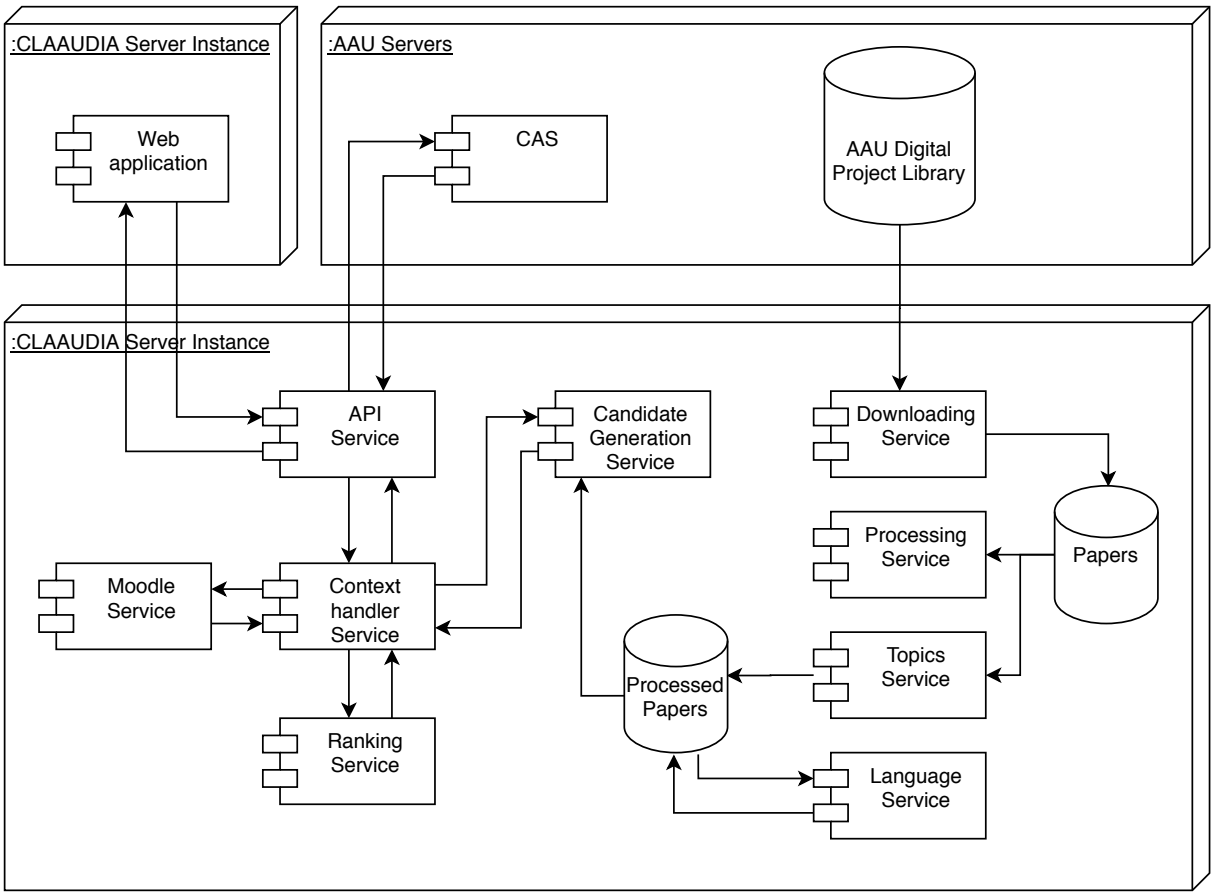

Figure 19: Deployment diagram of all internal and external entities involved with the system.

6.2.2 Online back-end services

Figure 21 shows a sequence diagram of the online services that respond to requests from the user through the user application. The first semi-online step is initialization of the model TF-IDF model in the candidate service, where it pulls the processed and prepared data from local storage. A user accesses the web application and makes a request for recommendations to the API, with a course input. The API ensures that the user is authenticated, before forwarding this request to the context handler. The context handler recognizes the input as a Moodle course and contacts the Moodle service to resolve it. The Moodle service authenticates to Moodle, access the course page and retrieves the content of the course, which it returns to the context handler. The course input is now treated as any other input, and sent along to the candidate generation service, where its used to find similar papers. A list of papers are returned to the context handler, and the context handler immediately forwards this list to the ranking service, which rearranges the list. The list is returned through the context handler, through the API to the user application, where it can be presented to the user. It is practically impossible to avoid chaining the services and running them synchronously, as they depend

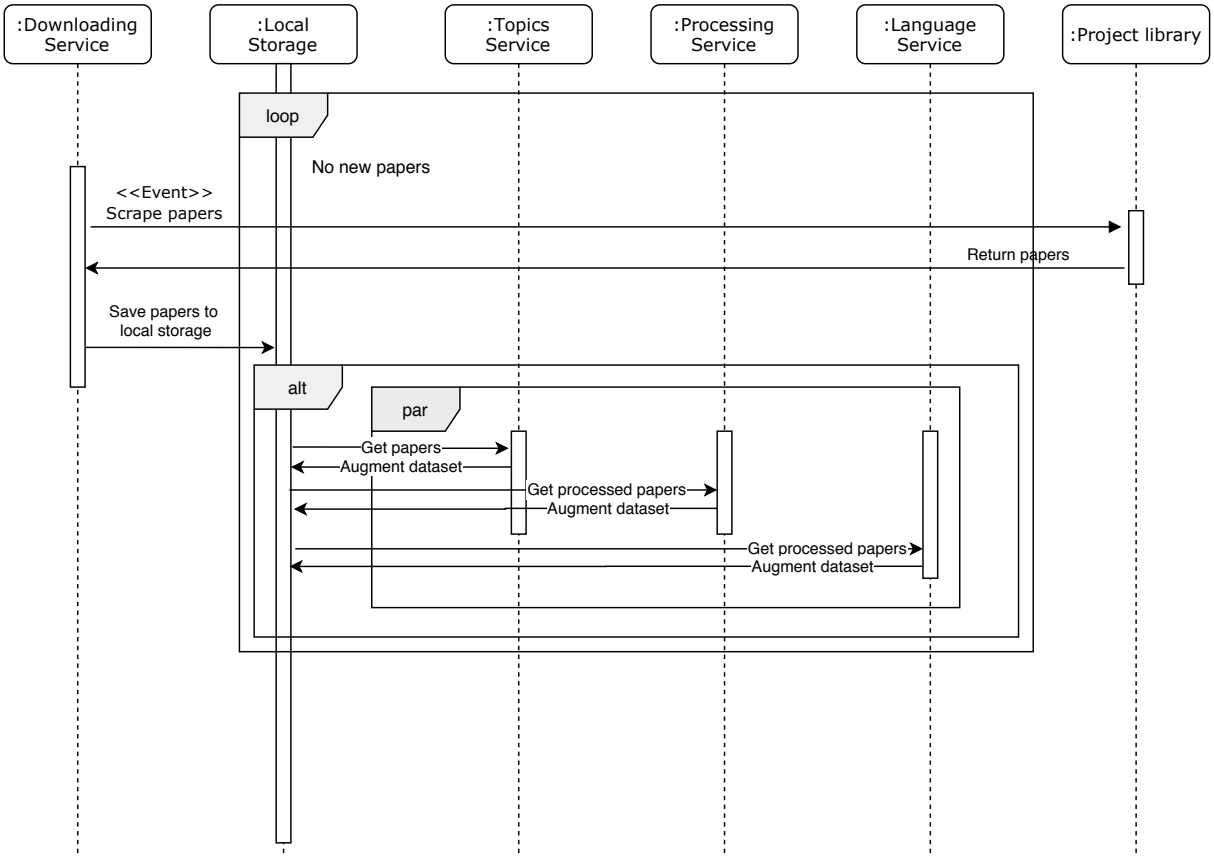

Figure 20: Sequence diagram of offline services.

on the output of a previous service. This goes against the MSA pattern and increases the response time on a request, since no processes are run in parallel. Common practice would be to replicate the data that is needed for a service from the service which has it, but since it is generated dynamically, this is also not possible. The culprit in terms of processing time, is the candidate generation, which could in theory be started asynchronously, but there would be nothing to show the user in mean time anyway. If the system employed HTTP polling and kept an open connection after making the request, the user application would receive notice that the processing was underway, while the user is shown a loading indicator. It is still not perfectly asynchronous, but at least the user application does not get suspicious that the request may be hanging.

While the context-handler facilities the communication from the API to the remaining services, it is not actively controlling them. They know what to do already and are not taking commands or instructions.

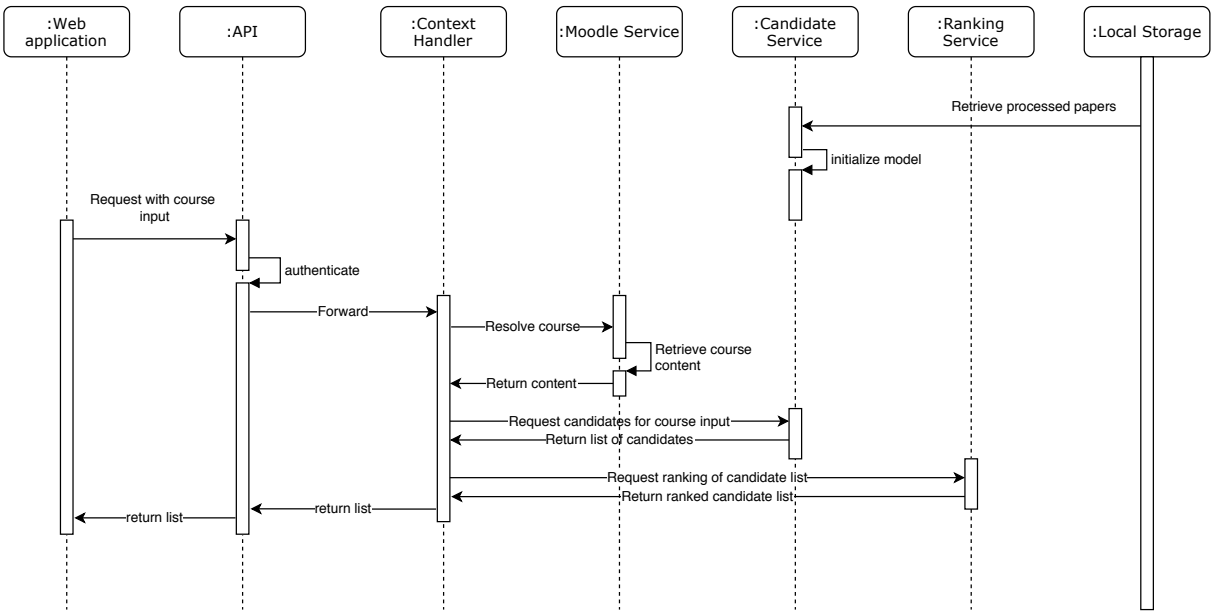

Figure 21: Sequence diagram of offline services.

7 Implementation

This section documents select parts of the implementation of the prototyped system. The system is vast and overflowing with functionality which is not directly relevant to the problem formulation, so the most significant parts of each components are presented.

7.1 User application

The user application is build as an Angular web application, which as mentioned previously, is a framework based on modularisation and reusability of components. This section will not go into detail about all the used components, but will focus on the searching, routing, presentation and remote server communication components. On the front page is a search component and a category component that shows the study boards. The search component (as seen in Figure 16, has an input field and a filtering panel. The input field is bound to the a model, using Angular's two way binding, which means any time the input changes in the interface, the variable value is immediately changed in the model, and vice versa. There is no need to read the input field explicitly when the value is needed, since the binding causes it to always be stored in the model. The same technique is used for the other dynamic information that is used in the search component, i.e. the total number of papers and the study boards to filter by. The user can press either the enter-button or the search icon, to execute the search method, which packages the search term and filters into a Filter object. The Filter object is encoded to base64 and made URL safe. These bytes are suffixed to a path, to which the web application is routed as seen

in Figure 22

```
126         const filterbytes = btoa(unescape(encodeURIComponent(JSON.stringify(this.filter))));
127         this.router.navigate( commands: ['/browse/search/' + filterbytes]);
```

Figure 22: Search routing

The routing component interprets the navigate command and finds a path template which match the one provided. In this case it matches one with ”browse/:action/:filterbytes”, which means there are two wildcards. One for the action and one for the data to perform that action to. This path activates the presentation component, that replaces the category component, while the search component stays in place. The same procedure took place when the user requested the front page and the base path components of the application were activated. The first time these component were activated, the router made sure data from remote servers, which they depended on was resolved and each of these components have subscribed to the resolver, so they can fully activate when the data is there. The presentation component receives the data from a shared service, where the search component saved the search results and filter. Meaning it takes the array from the shared service and displays it in the format shown in Figure 17. The results are in JSON format, with keys denoting the attributes of the papers, which can be presented on the cards. Further, the filter is used, to highlight the matching search terms on the cards (Figure 23). Angular has an augmented HTML language, which allows looping over model objects and creating instances of elements for each. The model object is the results array, which creates a card for each paper and binds each attribute to an element on the card. Lastly, the remote

```
<div  class="card-info">
  <div>
    <p style="font-size: smaller" [innerHTML]=""> <b> Publishing year:</b> {{card.publishingYear}}<br>
      <b>Education:</b> {{decodeURL(card.education)}}<br> <b>Semester:</b> {{card.semester}}</p>
    <button  mat-stroked-button (click)="searchForPaper(card)">Search library for similar papers -> </button>
  </div>
  <div style="text-align: right">
    <a  href="http://130.226.98.133:7000/{{card.path}}" target="_blank">View full paper</a>
    <div>
      <br><p style="font-size: smaller"> Pages: {{card.pages}}</p>
      <mat-chip-list >
        <div *ngFor="let word of card.keywords.split( separator: ',')">
          <mat-chip>{{word}}</mat-chip>
        </div>
      </mat-chip-list>
    </div>
  </div>
```

Figure 23: Configuration of card subtitle info

server requests to the API service, have to be protected by authentication. They can be protected on the server side, but the user application must also be configured to handle the tokens, to ensure easy usability. The alternative is that the users go and authenticate themselves. Accessing the application,

prompts several API calls, the first of which requests the total number of papers in the library, which is shown in the input field. This field is protected by CAS on the API, which tells the user application to redirect the user to the CAS login page at AAU, to get authenticated. When authenticated, the user is returned to the front page of the user application, with a ticket granting ticket that represents the SSO session, which the application can send to the API.

7.2 Services

All the online services are built as Flask [69] applications. Flask is a web framework, which is lightweight and ideal for microservices, due to its simplicity and anti bloating principles where there are no required libraries or tools. Flask offers an extremely fast and easy way to run a web server using Python, which can be scaled for more complex applications. Each services runs internally on one of the CLAAUDIA servers, where each is assigned a port number, from which it can offer its endpoints. Working with many remote services and manipulating them through SSH connection can be difficult to manage and with a MSA, multiple process also have to run concurrently, to support all the features. Running a process through an SSH connection can occupy the terminal, thus not allowing other services to be run through this connection. The tool TMUX [70], is a terminal multiplexer, which allows multiple terminals to be managed through the same SSH connection. Further, it maintains the connections of these, preventing them from timing out from inactivity.

7.2.1 Download

This first service downloads the entirety of the project library, such that it is locally available, rather than downloading on demand. It replicates the hierarchy of the library, to keep order of the papers and persist information about each paper's association. It is built as a web crawling bot, which is based on a web driver, able to run headlessly on CLAAUDIA. The first step, because papers other than master's these are protected, is to authenticate to AAU services. Login credentials are put in and submitted, and the cookies from this session are saved in the bot and sent along for all following requests. Getting the papers, first requires getting a list of all the education's id and using the REST service to visit the education's page one at a time. For each education page, the list of papers is expanded and if the number surpassed 500, the pages were also traversed. For each paper, the paper clip is pressed, initiating the download. Using tika, the paper is downloaded and saved lowest in the hierarchy structure of department, study board, education, paper. Added onto this service is the meta-data downloading, which scrapes the attributes of the paper off the library, including the semester, publishing year and title. This information is added directly to the PDF documents, using the PDFPY2 library that allows read and write of PDF and XML.

7.2.2 Process

The processing service deals with the extraction and cleaning of paper attributes from the PDF. This implementation only extract the abstract section from the papers. Using the things discovered about abstract formats, an extraction method is built, using tika to parse the PDFs as XML documents. When in XML format, the content text content can be extracted easily and the page separation is indicated. The content is split by page and iterated over. For each page, the extractor looks for an indication of an abstract, based on a list of synonyms for abstract and other possible identifiers, such as those pertaining to the formal title page, where there is always an abstract. If there is an abstract, the whole page is sent along. If the abstract was found on a title page, the known noise of the title page is filtered out. This abstract is processed using the researched pre-processing techniques, including converting the text to lower case, removing punctuation, removing apostrophes, removing English, Danish and custom stop words, converting numbers into words, stemming and finally lemmatizing the words. This processed version of the abstract is saved locally along with the original abstract and the other attributes extracted from the PDF, i.e. the title and other meta data written by the downloading service.

7.2.3 Topics

This service generates keywords for each paper that represent the content of the paper. The topics are based upon the same TF-IDF calculations done for candidate generation, as this technique is built to find words which are common in the paper but uncommon between papers. Here, the calculations are a little simpler as it is a typical implementation of Tf-IDF that can be done with just a couple of lines of code, using a prebuilt library (see Figure 24. In this case, two classes from the sklearn [**sklearn**]

```
145         # configure vector
146         cv=CountVectorizer(ngram_range=(1, 2))
147         # generate tdidf word counts
148         word_count_vector=cv.fit_transform(docs)
149         tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True)
150         tfidf_transformer.fit(word_count_vector)
151         new_dataset = []
152         for i in range(len(dataset)):
153             print(i)
154             doc = dataset[i]
155             # count matrix
156             count_vector=cv.transform([docs[i]])
157             # tf-idf scores
158             tf_idf_vector=tfidf_transformer.transform(count_vector)
```

Figure 24: TF-IDF vector calculations for the topics service

feature extraction module are used, namely TfidfTransformer and CountVectorizer. A CountVectorizer instances is created, configured to create 1- and 2-gram tokens, since keywords, when generated automatically often are not very descriptive and more often entity names. Thus, with compound words some of the local positioning is preserved, meaning adjectives and qualifiers are added to entities to create more meaningful keywords. A word count vector is created by transforming an array of all

the paper's full content into a word count vector using CountVectorizer. This whole vector is fitted to an instance of the TfidfTransformer class, such that word count is known for all words in the whole corpus (IDF). For each paper, a word count vector is created containing the counts for the particular paper (TF), and using the TfidfTransformer instance, the TF-IDF vector is created for the paper. from that vector, the top five tokens are selected and a string is composed of these, which is saved to the local dataset.

7.2.4 Language

This service was built for problem analysis, to detect the language of all papers in their full length, however it was adjusted for the system, to detect based on the abstracts only. Which is much simpler and avoids having to sample different parts of the paper to get an accurate results. The caveat here, is that some Danish papers have English abstracts, which means they will be labeled correctly according to the abstracts shown on the cards, but not the full paper. The language service load new processed papers from the local storage and adds a key to the dataset denoting "en", "da" or "other" for each paper, based on language detection using the langdetect library.

7.2.5 API

This is the entry point for the service and responsible for authentication. Authentication is implemented with CAS, using the Flask library, flask-cas [71], which abstract a lot of the authentication steps, i.e. are users already authenticated, do they have a ticket granting ticket to be verified or should they be directed to a CAS server to authenticate themselves. Requests are received from the user application and assuming a service ticket is passed along, it can request a proxy granting ticket from CAS, which can be used to authenticate the API during the user application's authenticated session. Four

```
32  # configure cas
33  app.secret_key = 'key here'
34  cas = CAS(app)
35  app.config['CAS_SERVER'] = 'https://signon.aau.dk/'
36  app.config['CAS_LOGIN_ROUTE'] = '/cas/login'
```

(a) Configuration of CAS

```
83  # get the number of papers indexed
84  @app.route('/getpapers', methods=['GET', 'POST'])
85  @login_required
86  def getTotalPapers():
87      return requests.get(contextHandlerUrl+"getpapers").text
```

(b) Endpoint protection with CAS

Figure 25: API CAS implementation

endpoints are offered to authentication users. Get the total number of papers, get the names of study boards, get the PDF of a paper and get a recommendation based on a filter. This services handles none of the logic for these, but instead forward them the the context handler. Figure 26 shows how the Flask service is started on the local IP-address of the machine, running on the port 7000. The other online services are started in the same way, on the local IP-address, however the are assigned other ports in the 7001-7004 range. The API service runs without SSL during development, but since it communicates with the outside, it should have it enabled for deployment.

```
if __name__ == "__main__":
    print("main called")
    #HTTPS
    #app.run(host='0.0.0.0',port="7000",debug=True,ssl_context=('cert.pem', 'key.pem'))
    #HTTP
    app.run(host='0.0.0.0',port="7000",debug=True)
```

Figure 26: Initiation of the API Flask service

7.2.6 Context handler

The job of the context handler is to direct the communication to the services in its cluster, meaning it has to know the locations of each of those services. It has endpoints corresponding those of the API. When a request for a recommendation is forwarded here, the context handler determines whether the search term is a text string or a Moodle course. If the input is a Moodle course, i.e. it contains the "moodle.aau.dk" string, it sends along the course link to the Moodle service and awaits response, which is used to call the candidate generation endpoint on the candidate generation service. The list received is sent along to the next service, the ranking service.

7.2.7 Moodle

Based on a course link, this service has to retrieve the textual content of that course. It does so by first authenticating with Moodle, the same way the Downloading service does, since they both use CAS authentication. Using the session cookie, it sends a get request to the course page and scrape the returned HTML content for the "region-main" tag, which includes the main content of the page, but avoids the calendars and headers of the page. Further, the breadcrumbs on the page are used to deduce which study board the course belongs to. In the implementation, it ended up not being used, but the idea was to increase the relevancy score of papers from the same study board. Both the course content and the study board it belongs to are returned in an array to the context handler.

7.2.8 Candidates generation

All the data has already been prepared for this service, thus the first step is to read the processed paper dataset files in from local storage, as they are going to be used to initialize the TF-IDF model in memory. A model is initialized for each study board without using any libraries. A manual implementation of TF-IDF is made in order to customize the weights given to words found in the paper title. Unlike the TF-IDF calculations made in the topics service, this is not done with a few lines of code, this is quite lengthy computations with complex matrix operations, filling more than a hundred lines of code. The whole point of this is to do particular weighing of title words which is shown in Figure 27, where all terms are multiplied by a value of 0.3, after which terms that are in the title are replaced by their original TF-IDF value. When a recommendation request arrives, the input goes through the

```
286         # for each word in the vector
287         for i in tf_idf:
288             tf_idf[i] *= alpha
289         # for each word in the title vector
290         for i in tf_idf_title:
291             tf_idf[i] = tf_idf_title[i]
```

Figure 27: TF-IDF vector calculations for the candidate generation service

same pre-processing as the papers did to convert it into a similar shape, with the same term forms. The cosine similarity is calculated for each of the other paper TF-IDF vectors from the selected study boards. Assuming the similarity is higher than 0, i.e. at least one word matches, the paper is added to a list along with its similarity measurement (relevance score) and considered a candidate.

7.2.9 Ranking

This service receives the candidate list from the context handler and the first thing it does is sort candidates based on the relevance score. Knowing which are most relevant allows the service to find the most common study boards and educations in a top ten selection. The most relevant boards and most relevant educations are used to increase the relevance scores of all the candidates as seen in 28. If a paper is in the same study board as the most relevant one, 0.1 is added to the relevance score. If a paper is in the same education as the most relevant one, 0.2 is added to the relevance score. Additionally, the paper is rewarded for recency and papers get their relevancy score increase proportionally to how recent they are. The numbers for value increase are arbitrarily chosen, but tweaked to get best results.

```
topBoard = max(studyboard, key=studyboard.get)
topEducation = max(educations, key=educations.get)
for i in range(len(papers)):
    #Weigh based on board
    if papers[i]['studyBoard'] == topBoard:
        papers[i]['relevance'] = float(papers[i]['relevance']) + 0.1
    #Weigh based on education
    if papers[i]['education'] == topEducation:
        papers[i]['relevance'] = float(papers[i]['relevance']) + 0.2
    #Weigh based on publishing year
    try:
        papers[i]['relevance'] = float(papers[i]['relevance']) + ((int(papers[i]['publishingYear']))-2000)/100
```

Figure 28: Increase in relevance score based on paper attributes

8 Testing

This section covers different types of software testing to ensure that requirements are properly implemented and that the system is stable. Requirements implementations are tested by both unit tests and end-to-end tests. Both are necessary to cover individual functions and key user experience flows. The tests are based on automated tests, with the purpose of continuous integration and continuous delivery.

8.1 Unit testing

Tests are written for each back-end unit using the Python test framework unittest [72], while Angular brings its own framework Karma. The goal of the unit tests is to assert that inputs to unit produce the expected output. There are a large number of units in the system and therefore a general unit test class is created in both frameworks, which the unit specific tests can inherit from that contains methods to set up and tear down the service. This is important, to guarantee each service is tested in solitude and a test is not affected by another running test. Each unit test inherits from the base class, and then implements its own methods to test the unit. The units are one of the three levels of granularity: a module, class or a method. This section is not going to detail the testing of every module, class and method, rather it will focus on some that are of crucial value to the system. Furthermore, far from every unit has been tested in this system. The service that got the most benefit from testing was the candidate generation service, because required a lot of tweaking to get right. The unit tests for the service are based around the HTTP endpoints it makes available to the other services, including candidate generation from input, random candidate generation, total number of papers and names of the study boards. The endpoints are tested by calling it with an input and asserting equivalence between what is returned and what is expected. Figure 29 shows the unit the for candidate generation based on an input, where the input consists of an array of selected study boards and a search term. For this particular test, only one board is selected and a Moodle course is used as input. To assert the success of this test, the method must return a two-part array, where the datatype of the title of the 0th element of the 1st element in that array must be a string. This confirms that what was returned contains some legitimate data. For this test, replacing the input is as simple as changing the payload variable. The second test of this service, test the unit which returns the total number of papers. This unit does not require an input and the endpoint can be called with a HTTP GET request. For this test to be successful, the unit must return an integer value. The expected value is not specified, as this is bound to change. What is important is that it returns a non-zero integer value. Getting the correct value is the job of another unit, within this unit. This smaller unit, the ”countPapers” method has to produce the correct number of papers and since this method is not an endpoint, the function has to be imported to the test suite from the service. Since the method has a dependency, i.e. the data it counts,

```python
class TestAPI(BaseCase):
    def test_getrecfilter(self):
        print("current test: test_getrecfilter")
        # Given
        payload = json.dumps({
            "studyBoards": ["Study Board of Electronics and Information Technologytest.npz"],
            "searchTerm": "https://www.moodle.aau.dk/course/view.php?id=29492"
        })
        # Call self
        response = self.app.post('/getrecfilter', headers={"Content-Type": "application/json"}, data=payload)
        data = json.loads(response.data)

        #Two part array
        self.assertEqual(2, len(data))

        #Contains objects with title property
        self.assertEqual(type(data[1][0]["title"]),str)
        self.assertEqual(200, response.status_code)
```

Figure 29: Unit test case for candidate generation based on input

this dependency has to be stubbed in the test suite, to emulate the real data. Now, it can be tested with whichever input, and the output can be compared with the expected result. It goes even deeper, because there is another method which collects the real data, but that is for another time. Moving on to another service, the language detection service, which was heavily used in problem analysis and became necessary to augment the dataset with language attributes for papers. The method being tested takes processed paper data as input, verifies that language is not set already and adds it if missing. Figure 30 shows the unit test for it, where a dataset is mocked with an English abstract, which the unit is expected to recognize as English, and augment the dataset with the ”lang” attribute value ”en”. With a list of test cases for each service, it is easy to run them all at once, after changes have been made

```python
class TestLanguageService(BaseCase):

    def test_augmentDataset(self):
        dataset = json.dumps({[
            "title":"Title of the paper",
            "abstract":"An abstract which is clearly in English"}
        ]})
        augmentedDataset = json.loads(augmentDataset(dataset))
        self.assertEqual(augmentDataset["lang"],"en")
```

Figure 30: Unit test case language detection

in the system. Ideally, this would occur automatically, but for now, they are triggered manually via an SSH terminal connection to the server they reside on.

8.2 End-to-end testing

The end-to-end tests also include both the user application and the online services of the system. The requirements being tested via the user application depend on the online services to work just as how their requirements dictate. The end-to-end tests are written using the tool Protractor, which is integrated in Angular. The tool is based on selenium and interacts with the system through a browser,

to emulate the behaviour of a user. Using protractor with Angular, makes locating objects easier as it offers Angular specific strategies, such as locating elements based on data models. The test here are rather superficial as they have to account for dynamic content, thus the tests rely more on asserting that elements exist, rather than their exact value. Unless the value is static, tests do not have to be so specific that they have to be rewritten every time an input changes. Following is a list of the automatic tests that are performed in order to assure the requirements for the system are implemented:

- **FR#1, FR#8, FR#9, FR#10, FR#12:** Asserts that a Moodle course can be used as input, by setting the value of the input field to a link to a course, sending an ”Enter” key and checking if the candidates are shown on the /browse path, by counting elements with the ”.mat-card” CSS tag. This also means CAS and Moodle authentication was successful and course content was retrieved.
- **FR#2:** Asserts that a string input can be used as input, by setting the value of the input field to a string, sending an ”Enter” key and checking if candidates are shown the /browse path, by counting elements with the ”mat-card” CSS tag.
- **FR#3:** Asserts that a paper input can be used as input, by performing a search, pressing the ”search library for similar papers”-button and checking if new candidates are generated.
- **FR#4:** Asserts that a filter can be applied to a search, by specifying a study board for a search, and verifying that the cards show all have that study board value.
- **FR#5, FR#18, FR#19:** Asserts that recommendation results are shown with information on the paper, by locating ”mat-card” elements and checking that the attributes and pertaining values exist.
- **FR#6:** Asserts that a list can be sorted by year, semester and relevance, by locating the select element through CSS and selecting each of the options, starting from the second and ending on relevance again, and checking that the list changes order. The order change is detected by having different top two card titles.
- **FR#7:** Asserts that the category/study board cards exist on the front page, by locating all objects with the CSS class ”.mat-card-content” on the page, and compare the length of that list with the expected number of active study boards.
- **FR#14:** Asserts that random candidates are generated by clicking a category/study board card, and checking that papers are shown on the /browse path, by counting elements with the ”.mat-card” CSS tag.

There are a few implemented requirements which cannot be tested in this manner and have to be addressed in other ways. FR#15 state that the service must rank the selected candidates, however the

user application does not know how the list was arranged before there ranking and cannot compare. Therefore, this ranking has to be tested by deactivating the ranking module and comparing the list of results before and after. There is no automatic way to do this in the system and it has to be done manually. The same goes for requirements FR#16, FR#17 and FR#20, where their requirement fulfilment cannot be guaranteed based on the web application interactions alone - fortunately, the unit tests can guarantee their function.

9 Discussion

There have been many elements to this project and this has resulted in some of them being rather superfluous, which means there are many further avenues to pursue, in terms of improving the system on all fronts, provided the time. This discussion touches on some aspects of the project, which caused issue or where more research could be done. The first thing to ask, after the fact, is whether this system is a recommendation system or an IR and IF system. The main differences according to literature, is the usage of natural language input from the user in IR, where recommender systems usually take no explicit inputs. There is an argument to be made that since an input is required, which typically consists of natural language, as a user defined query or indirectly as a course page, that the system is more-so an IR system. On the contrary, the intention behind the course page inputs are to interpret them as a representation of user interest, not unlike a user profile, which is a widely used feature for recommender systems. In a way, temporary profiles are created, that specify a set of interests in the moment.

The efforts put into analysing the project library in terms of language and length, went largely to waste. The primary output from those analyses was the level of diversity among different groupings of papers, but not much in terms of utilizing the vastness of content in student papers. Despite the dominant tendencies to use only title and abstracts content from academic papers, it is believed that more of the information could be used. The textual content is sometimes overwhelming with these long papers, however, with some sophisticated NLP it could be possible to further improve the representation of the paper. It has been widely proven that longer and more descriptive queries increase recommendation accuracy, thus it should be possible to boil down a student paper to its most descriptive features and end up with a passage longer than the abstract, which is even closer to the true identity of the paper.

The experiments showed that accuracy depended on document shapes and input lengths, but it is also showed that there are large differences between the accuracy of recommendation in different domains. The IAM inputs performed consistently better, presumably because the domain is narrow and projects are usually along the same lines, whereas the IoT domain holds more diverse project topics. Additionally, there are domains, where similar papers are just far and few between, making it difficult to present a long list of accurate results. Academic research often concerns new topics and domains that have not been researched a lot before, which also are difficult to find recommendations for. As far as accuracy goes, there was not much testing in the way of perceived accuracy, which according to state of the art research was sometimes more important than objective evaluations. There were no online tests or user studies with the goal om improving ranking of candidates. Such evaluation could have huge impact on the accuracy, but it would take time, before enough data could be collected, and it would require consistent users for the system. For the moment, the recommendations have

no relation to users or their history, which could be considered a positive trait, as the there are no prejudices and the recommendations are not based on the opinions of anyone else. Arguably, it is not based on the user's opinion either. Generally, since the driving force of the project was so user-centric, i.e. what odes it take to keep users coming back to the service and what do they want in such a service, there could have been more inclusion. Quantitative and qualitative interviews about the potential interest in such a system more open discussions of how users are encouraged to do anything.

Building the system with a MSA was an early design decision and while the system benefited from it in many ways, it also turned out to not be the perfect fit for all situations. Situations where multiple services are dealing with the same data stream, the architecture will have to morph into a hybrid shape, where services are asynchronous, from the perspective of the user application, such that threads are not blocked, but using HTTP polling, the system still waits for a result before the next thing happens.

During this project, the microservices were managed through terminals with tmux, however for a stable system there would have to be a rigid user interface to deal with status of the service, where they can be monitored, disabled restarted and so on. Running nine services in nine different terminals clouds the overview.

Lastly, as the system was never meant to be a stand-alone recommendation system, but rather a plugin service for other platforms, it still tickles the mind whether this service would see as much use as one that was integrated into a site such as Moodle, where the students are caught right in the moment of their peak interest for a course, and completely primed to receive recommendations pertaining to it. It is likely it would have more eyes on it, but there would of course be limitations as the users could not customize their recommendation prompt.

10 Conclusion

The purpose of this project was to design and build a service to deal with the shortcomings of the AAU Digitial Project Library. To lead the research direction towards that purpose the following problem formulation was defined:

"How can an information retrieval system be utilized to encourage students' usage of AAU's Digitial Project Library?"

There was plenty of room for improvements with the existing library as the implemented IR mechanism are extremely simple. The analyses and design process show a potentially viable solution in a IR-recommendation system hybrid, which is based on in-the-moment inputs that carry a lot of weight in terms of indicating the users interest. It was found that a system of this type, which has the innate quality of never truly being completed, should be built flexibility in mind, so improvements are easy to make in the future, without rebuilding the system. The following sub-questions highlight some key goals of the project, which contribute towards the main problem.

"How is sufficient input conveniently provided?"

"What should be retrieved by the system to satisfy a student's input"

"How should a user interact with the system, to receive its value"

The first sub-question lead to exploration of Moodle courses as inputs for recommendations. They are familiar to students and represent areas of interest, which are shared by students at one point in time. While the sample size was too small, course input showed promise in being able to retrieve relevant papers for some domains, while others were unsuccessful, leading to the conclusion that some areas are harder to conjure recommendations for. The second sub-question mandates investigation into why a paper should be considered relevant and different evaluation metrics for recommendation systems. As a start, objective offline evaluation metrics will suffice to asses the accuracy of generated candidate papers to a student's input - in the future more user involvement in evaluation might give other results. The third sub-question was answered by way of user testing, where multiple iterations of prototypes were put in the hands of potential users, eventually leading to a satisfactory user experience for the majority of the subjects. Improvements are still possible in almost all regards of this project, but the research, analysis, design and implementation lead to a proof-of-concept that is more effective and particular in generating search results than the current Digital Project Library.

References

[1] *langdetect*. https://github.com/Mimino666/langdetect. Accessed: 04-08-2020.

[2] *Association for Computing Machinery*. https://www.acm.org/. Accessed: 04-08-2020.

[3] *The world's largest technical professional organization for the advancement of technology*. https://www.ieee.org/. Accessed: 04-08-2020.

[4] Jakob Nielsen. *Thinking Aloud: The #1 Usability Tool*. Jan. 2012. URL: https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/.

[5] Jakob Nielsen. *Why You Only Need to Test with 5 Users*. Mar. 2000. URL: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

[6] Jakob Nielsen and Thomas K. Landauer. "A Mathematical Model of the Finding of Usability Problems". In: *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*. CHI '93. Amsterdam, The Netherlands: Association for Computing Machinery, 1993, pp. 206–213. ISBN: 0897915755. DOI: 10.1145/169059.169166. URL: https://doi.org/10.1145/169059.169166.

[7] *Moodle*. https://moodle.org/. Accessed: 04-08-2020.

[8] Joeran Beel et al. "Research Paper Recommender Systems: A Literature Survey". In: *International Journal on Digital Libraries* (2015), pp. 1–34. ISSN: 1432-5012. DOI: 10.1007/s00799-015-0156-0.

[9] Sarah Hatton. "Early Prioritisation of Goals". In: vol. 4802. Nov. 2007, pp. 235–244. DOI: 10.1007/978-3-540-76292-8_29.

[10] Bogdan Vasilescu et al. "Quality and Productivity Outcomes Relating to Continuous Integration in GitHub". In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2015. Bergamo, Italy: Association for Computing Machinery, 2015, pp. 805–816. ISBN: 9781450336758. DOI: 10.1145/2786805.2786850. URL: https://doi.org/10.1145/2786805.2786850.

[11] *Git*. https://git-scm.com/. Accessed: 04-08-2020.

[12] Mari Vallez and Rafael Pedraza-Jimenez. "Natural Language Processing in Textual Information Retrieval and Related Topics [on line]". In: *Hipertext.net* num.5 (May 2007).

[13] Giambattista Amati. "Information Retrieval". In: *Encyclopedia of Database Systems*. Ed. by Ling Liu and M. Tamer Özsu. New York, NY: Springer New York, 2018, pp. 1970–1975. ISBN: 978-1-4614-8265-9. DOI: 10.1007/978-1-4614-8265-9_915. URL: https://doi.org/10.1007/978-1-4614-8265-9_915.

[14] W. Bruce Croft, Donald Metzler, and Trevor Strohman. "Search Engines - Information Retrieval in Practice". In: 2009.

[15] Xiaomei Bai et al. "Scientific Paper Recommendation: A Survey". In: vol. PP. Jan. 2019, pp. 1–1. DOI: 10.1109/ACCESS.2018.2890388.

[16] Felice Ferrara, Nirmala Pudota, and Carlo Tasso. "A Keyphrase-Based Paper Recommender System". In: vol. 249. Jan. 2011, pp. 14–25. DOI: 10.1007/978-3-642-27302-5_2.

[17] A. I. Kadhim, Y. Cheah, and N. H. Ahamed. "Text Document Preprocessing and Dimension Reduction Techniques for Text Document Clustering". In: *2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology*. 2014, pp. 69–73.

[18] Matthew Lyle Olson, Lisa Zhang, and Chun-Nam Yu. {*ADAPTING*} {*PRETRAINED*} {*LANGUAGE*} {*MODELS*} {*FOR*} {*LONG*} {*DOCUMENT*} {*CLASSIFICATION*}. 2020. URL: https://openreview.net/forum?id=ryxW804FPH.

[19] Zichao Yang et al. "Hierarchical Attention Networks for Document Classification". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 1480–1489. DOI: 10.18653/v1/N16-1174. URL: https://www.aclweb.org/anthology/N16-1174.

[20] Shaobo Li et al. "DeepPatent: Patent Classification with Convolutional Neural Networks and Word Embedding". In: *Scientometrics* 117.2 (Nov. 2018), pp. 721–744. ISSN: 0138-9130. DOI: 10.1007/s11192-018-2905-5. URL: https://doi.org/10.1007/s11192-018-2905-5.

[21] Lulu Wan et al. *Long-length Legal Document Classification*. 2019. arXiv: 1912.06905 [cs.CL].

[22] Damien Hanyurwimfura et al. "An Effective Academic Research Papers Recommendation for Non-profiled Users". In: vol. 8. Mar. 2015, pp. 255–272. DOI: 10.14257/ijhit.2015.8.3.23.

[23] Simon Philip, Peter Shola, and Ovye Abari. "Application of Content-Based Approach in Research Paper Recommendation System for a Digital Library". In: *International Journal of Advanced Computer Science and Applications* 5 (Oct. 2014). DOI: 10.14569/IJACSA.2014.051006.

[24] Yeon-Chang Lee et al. "Recommendation of research papers in DBpia: A Hybrid approach exploiting content and collaborative data". In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2016, pp. 002966–002971.

[25] *HThe ABCs of Learning to Rank*. https://lucidworks.com/post/abcs-learning-to-rank/. Accessed: 04-08-2020.

[26] *Search at Slack*. https://slack.engineering/search-at-slack-431f8c80619e. Accessed: 04-08-2020.

[27] *Learning to Rank for Flight Itinerary Search*. https://hackernoon.com/learning-to-rank-for-flight-itinerary-search-8594761eb867. Accessed: 04-08-2020.

[28] *How We Use Machine Learning and Natural Language Processing to Empower Search*. https://tech.wayfair.com/2018/10/how-we-use-machine-learning-and-natural-language-processing-to-empower-search/. Accessed: 04-08-2020.

[29] *Apache Lucene - Scoring*. https://lucene.apache.org/core/3_5_0/scoring.html. Accessed: 04-08-2020.

[30] V. U. Thompson, C. Panchev, and M. Oakes. "Performance evaluation of similarity measures on similar and dissimilar text retrieval". In: *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*. Vol. 01. 2015, pp. 577–584.

[31] Sean M. McNee et al. "On the Recommending of Citations for Research Papers". In: *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work*. CSCW '02. New Orleans, Louisiana, USA: Association for Computing Machinery, 2002, pp. 116–125. ISBN: 1581135602. DOI: 10.1145/587078.587096. URL: https://doi.org/10.1145/587078.587096.

[32] Paolo Cremonesi, Franca Garzotto, and Roberto Turrin. "Investigating the Persuasion Potential of Recommender Systems from a Quality Perspective: An Empirical Study". In: *ACM Trans. Interact. Intell. Syst.* 2.2 (June 2012). ISSN: 2160-6455. DOI: 10.1145/2209310.2209314. URL: https://doi.org/10.1145/2209310.2209314.

[33] Bart Knijnenburg et al. “Explaining the user experience of recommender systems”. In: *User Modeling and User-Adapted Interaction* 22 (Oct. 2012), pp. 441–504. DOI: 10.1007/s11257-011-9118-4.

[34] Katrien Verbert et al. “Visualizing Recommendations to Support Exploration, Transparency and Controllability”. In: *Proceedings of the 2013 International Conference on Intelligent User Interfaces*. IUI ’13. Santa Monica, California, USA: Association for Computing Machinery, 2013, pp. 351–362. ISBN: 9781450319652. DOI: 10.1145/2449396.2449442. URL: https://doi.org/10.1145/2449396.2449442.

[35] Pasquale Lops et al. “MARS: a MultilAnguage Recommender System”. In: (Oct. 2). DOI: 10.1145/1869446.1869450.

[36] Fedelucio Narducci et al. “Concept-Based Item Representations for a Cross-Lingual Content-Based Recommendation Process”. In: *Inf. Sci.* 374.C (Dec. 2016), pp. 15–31. ISSN: 0020-0255. DOI: 10.1016/j.ins.2016.09.022. URL: https://doi-org.zorac.aub.aau.dk/10.1016/j.ins.2016.09.022.

[37] Osmar Castrillo-Fernández. “Web scraping: applications and tools”. In: *European Public Sector Information Platform* (2015).

[38] *Beautiful Soup Documentation*. https://www.crummy.com/software/BeautifulSoup/bs4/doc/. Accessed: 04-08-2020.

[39] *Selenium Browser Automation*. https://www.selenium.dev/. Accessed: 04-08-2020.

[40] *PyPDF2 Documentation*. https://pythonhosted.org/PyPDF2/. Accessed: 04-08-2020.

[41] *PDFMiner*. https://pdfminer-docs.readthedocs.io/pdfminer_index.html. Accessed: 04-08-2020.

[42] *Apache Tika - a content analysis toolkit*. http://tika.apache.org/. Accessed: 04-08-2020.

[43] M. H. Valipour et al. “A brief survey of software architecture concepts and service oriented architecture”. In: *2009 2nd IEEE International Conference on Computer Science and Information Technology*. 2009, pp. 34–38.

[44] F. Wang and F. Fahmi. “Constructing a Service Software with Microservices”. In: *2018 IEEE World Congress on Services (SERVICES)*. 2018, pp. 43–44.

[45] F. Rademacher, S. Sachweh, and A. Zündorf. “Differences between Model-Driven Development of Service-Oriented and Microservice Architecture”. In: *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. 2017, pp. 38–45.

[46] Gagan Murghai. “Microservice Orchestration Vs Choreography”. In: (Sept. 2019). URL: https://www.softobiz.com/microservice-orchestration-vs-choreography/.

[47] Samarpit Tuli. “Microservices vs SOA: What’s the Difference?” In: (May 2018). URL: https://dzone.com/articles/microservices-vs-soa-whats-the-difference.

[48] H. Kang, M. Le, and S. Tao. “Container and Microservice Driven Design for Cloud Infrastructure DevOps”. In: *2016 IEEE International Conference on Cloud Engineering (IC2E)*. 2016, pp. 202–211.

[49] M. Virmani. “Understanding DevOps bridging the gap from continuous integration to continuous delivery”. In: *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*. 2015, pp. 78–82.

[50] L. Chen. “Microservices: Architecting for Continuous Delivery and DevOps”. In: *2018 IEEE International Conference on Software Architecture (ICSA)*. 2018, pp. 39–397.

[51] L. Padró and J. Turmo. "TextServer: Cloud-Based Multilingual Natural Language Processing". In: *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. 2015, pp. 1636–1639.

[52] Google. *Natural Language*. 2020. URL: https://cloud.google.com/natural-language.

[53] IBM. *Watson Natural Language Understanding*. 2020. URL: https://www.ibm.com/cloud/watson-natural-language-understanding.

[54] Amazon. *Amazon Comprehend*. 2020. URL: https://aws.amazon.com/comprehend/.

[55] SYNCED. *The Staggering Cost of Training SOTA AI Models*. June 2019. URL: https://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/.

[56] Rowan Zellers et al. "Defending Against Neural Fake News". In: *Advances in Neural Information Processing Systems 32*. 2019.

[57] Google. *BERT*. Mar. 2020. URL: https://github.com/google-research/bert.

[58] NLTK. *Natural Language Toolkit*. 2020. URL: https://www.nltk.org/.

[59] AAU. *CLAAUDIA*. 2020. URL: https://www.claaudia.aau.dk/.

[60] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. USA: Cambridge University Press, 2008. ISBN: 0521865719.

[61] *Identity and Access Management (ICTEC8) (spring 2019)*. https://www.moodle.aau.dk/course/view.php?id=29489. Accessed: 04-08-2020.

[62] *Business Law (ITCOM6) (Spring 2018)*. https://www.moodle.aau.dk/course/view.php?id=25377. Accessed: 04-08-2020.

[63] Tomek Strzalkowski et al. "Evaluating Natural Language Processing Techniques in Information Retrieval". In: 1999.

[64] Hiteshwar Azad and Akshay Deepak. "Query Expansion Techniques for Information Retrieval: a Survey". In: (Aug. 2017).

[65] Wikipedia. *Wikipedia:Short description*. Mar. 2000. URL: https://en.wikipedia.org/wiki/Wikipedia:Short_description#Using_short_descriptions_in_Wikipedia.

[66] *Angular*. https://angular.io/. Accessed: 04-08-2020.

[67] *WAYF - Where Are You From*. https://wayf.dk/. Accessed: 04-08-2020.

[68] *CAS Protocol 3.0 Specificationm*. https://apereo.github.io/cas/5.1.x/protocol/CAS-Protocol-Specification.html. Accessed: 04-08-2020.

[69] *Flask*. https://flask.palletsprojects.com/en/1.1.x/. Accessed: 04-08-2020.

[70] *tmux*. https://github.com/tmux/tmux/wiki. Accessed: 04-08-2020.

[71] *Flask-CAS*. https://github.com/cameronbwhite/Flask-CAS. Accessed: 04-08-2020.

[72] *Unit testing framework*. https://docs.python.org/3/library/unittest.html#module-unittest. Accessed: 04-08-2020.

11 Appendix

11.1 Think aloud test 1

Task 1: Do a search for something you are interested in yourself.
Task 2: Find a course related to that interest on Moodle and use this content as the input

11.1.1 Test person: Mathias

C1: The paths for arriving at the value
Subject has no idea what the website is about or what is the stated recommendations are about? They assume you can find some papers about certain subjects. By searching for topics.

C2: The user experience of arriving at the value
The site is too white. Aesthetics are not pleasing. No indication of loading after pressing search. Or if there is a hold up on a server. Waiting time is not a problem if knowing when search is happening. Difficult time returning to front page. Tries pressing the logo. The part of the abstract shown is not relevant enough. To really find out what they are about, the full document has to be opened. The full caps text is not comfortable. Whole abstract would be nicer, but also might take too much space. Publishing year could be relevant. Authors not too relevant for students. Would be nice to have a filter for search results. Some search results labels for filters. Select how many search results.

C3: The perceived quality of the value
Subject recognizes that recommendation results are reasonably relevant and match some of the keywords they provided. Using a longer search query appeared to net more relevant results searching for the topic of Interaction Design.

C4: The ways that the value could be applied
Useful for literature search. You have a topic in mind, but don't know exactly how to approach it. Inspiration for own project. Kick-start to find direction for own project. Good to know how previous projects have been shaped. A problem is that no grades are available. There is no quality check on student papers. Won't be returning to the portal frequently, but might find interest when visiting a course page.

11.1.2 Test person: Christian

C1: The paths for arriving at the value
A system for finding student papers from search terms.You can search for papers and search for papers in study boards.

C2: The user experience of arriving at the value
Loading indicator is missing when searching. Text disappearing when trying to make a new search is annoying. Christian presses the logo trying to go to the front page. Abstract is almost useless and takes a long time to read. It does not look like an abstract on the card. Would like to have filtration options for language, year, categories (topics) and study boards. Would also like to find which board search terms match the best, to help narrow down search. Would like to find papers similar to results. Would like to see the page number on cards, to get an idea of size and comprehensiveness, as an indicator of quality (considering grades are not available). Would like as many results as possible, and the ability to

remove results after. Would like to go deeper in a search and stray into multiple directions by opening searches in a new tab and being able to save and return to a previous search.

C3: The perceived quality of the value
First search results seem relevant. There some top results that fit very well with what Christian was looking for. Search terms were ”Software Development”. Very fitting search results when searching for the Identity and Access Management course. Many relevant papers in top results. Poor results when searching for ICT in Organizations course. Few mildly relevant papers in top results.

C4: The ways that the value could be applied
When looking for inspiration for own project. Search could have description of own topic as input. In the beginning and during the project to get inspiration for different parts.

11.1.3 Test person: Kazi

C1: The paths for arriving at the value
No idea what the page is about. Perhaps to find something you want to study, since it is university related, the courses that you might want to take. You can search for something, or choose courses.

C2: The user experience of arriving at the value
Would like more information about process in papers. Abstract description is reasonable, perhaps show more of it. Perhaps show how many pages the paper is. Navigation is tough. No way back. Would like to filter more, based on language and courses.

C3: The perceived quality of the value
Results are not relevant to the search of ”micro economics”. There was a relevant results further down the page. A lot is danish or related to Denmark. A few relevant papers among first results when searching for ”ESB: Top-up Macro Economics” course.

C4: The ways that the value could be applied
To find out which course subject might want to take next semester. Or to supplement and learn more about a course topic.

11.1.4 Test person: Kathrine

C1: The paths for arriving at the value
The system recommends papers written by students. You can search among 20 thousand student papers or specify topic areas with the study board cards.

C2: The user experience of arriving at the value
Confusion about what the text on the cards is. Would like to see the whole abstract. Only useful thing is the title. Would like imagery to show something, like topics. Title should wrap, so the whole thing is shown. Would like more topical key words of the papers. Boards are not arranged logically. They could be grouped by subjects.

C3: The perceived quality of the value
Results looked somewhat relevant. Cross-language results were not as anticipated.

C4: The ways that the value could be applied
Useful when writing assignments for inspiration, by seeing others' approach to similar tasks.

11.1.5 Test person: Leyla

C1: The paths for arriving at the value
Searching among 20 thousand articles with keywords or search in categories of articles.

C2: The user experience of arriving at the value
Confused about the article fields. More clear labels are needed. Abstract is hard to read. More screen space should be used, to make room for information. Would like to know which semester and which courses the articles are linked to. Would like category tags and to filter on these tags.

C3: The perceived quality of the value
Results are relevant to information technological searches.

C4: The ways that the value could be applied
Useful for comparing with theses, to see methods of others as well as which citations are used in similar works.

11.2 Think aloud test 2

11.2.1 Test person: Mathias

Task 1: Do a search for something you are interested in yourself.
Task 2: Find a long paragraph related to that interest on the internet and use this content as the input for a search.
Task 3: Make another search and filter the search according to your needs.

C1: The paths for arriving at the value
Student paper recommender. Recommendations of papers written by students of the university. Users can explore study boards. Click them and see how many are under each subject. Search for something within them. Filter by language. Search for different things, such as author, topics, keywords.

C2: The user experience of arriving at the value
Subject feels the page should show number of results, to get an idea of the "success level" of the search. It is hard to see immediately how a paper is relevant. Abstracts are sometimes very long and contains uninteresting parts. Highlight the search terms in the texts. Searching for paper in library, should search whole library rather than only the study board of that paper. Searching in library indicates that the whole library is subject. Need search button, rather than enter because of filters list. It is too long and you end up far down the page. Would also like to filter by year. Filter by semester. Would like to filter study boards easier. There are many and it is difficult to find specific ones. Perhaps group into higher level categories.

C3: The perceived quality of the value
Search on "participatory design" gives what looks like generally relevant results. Search for part of Mathias' own thesis concerning this topic, also gave very relevant results.

C4: The ways that the value could be applied
Inspiration, literature search. Search for possible directions. Find information related to current project.

11.2.2 Test person: Christian

C1: The paths for arriving at the value
Filters are disorienting. Confusion whether a study board is a category. The filter needs to have a search button or option to apply filters to current search.

C2: The user experience of arriving at the value
Likes that abstract is optional. It needs to be easier to search again. Would like topics on the cards. Would like to persist notes about papers, and curate lists of papers for later. Would like the "search for similar papers" option to be more user-friendly in terms of the input.

11.2.3 Test person: Kazi

C1: The paths for arriving at the value
This time quickly realizes that list of study boards is scrollable and there are more options.

C2: The user experience of arriving at the value
Color choices seem more serious and professional. The text of the filter options is too small. Boards that have no papers in them, should be removed. There is no clear search button. Pressing the enter-button on from anywhere on the page should cause a search. Title on the cards does not say much the abstract is necessary.

11.2.4 Test person: Kathrine

C2: The user experience of arriving at the value
Surprised that results are relevant to stemmed version of input words. Input field is not very clear. It looks like a part of the header graphics. Filter looks like a search bar. Filter drop down looks like a new page. Filter drop down should have a search button. Would like to sort results, based on year and page number. Would like the boards to be organized better. Would like guidelines on how to make a search. Would like a clear path back to start. Whole card should be a link. Does not like colors or the logo.

11.2.5 Test person: Leyla

C2: The user experience of arriving at the value
Filter needs to include semester and year. There should be a button to select all study boards in the filter. Semester, study board and year should be on the paper cards, to see better how a result is relevant to a query. Would like to have suggestions when searching, i.e. auto-completion.

11.3 Think aloud test 3

11.3.1 Test person: Mathias

Task 1: Do a search for something you are interested in yourself.
Task 2: Make another search and filter the search according to your needs

C2: The user experience of arriving at the value
Would like to filter papers out after a search. Would like to force specific words to be a in the results. Would like to force combinations of multiple words.

11.3.2 Test person: Christian

C2: The user experience of arriving at the value
Highlight of text is confusing, because it only highlights parts of the matching words. Search mechanism is overdone, too much emphasis.

11.3.3 Test person: Kazi

C2: The user experience of arriving at the value
Filter is confusing as the clustering makes little sense. Would like to search for even more specific domains.

11.3.4 Test person: Kathrine

C2: The user experience of arriving at the value
Would like topics to only show adjectives when combined with another word. Sorting by semester is not relevant. Would like to have the magnifying glass inside the input field. Would like a better user experience, when there are no results.

11.3.5 Test person: Leyla

C2: The user experience of arriving at the value
Would like to change the layout of candidate paper cards into both rows and columns, to fit more into the view. Would like to have the abstract on a separate page, since it takes a click to get to anyway. Flags are too much. A little juvenile.

11.4 Data

11.4.1 Study boards

Board	Papers	Educations
Study Board of Computer Science	3611	28
Study Board of Planning and Surveying	3135	21
Study Board of Electronics and Information Technology	2881	61
Study Board of Communication and Digital Media	2601	19
Study Board of Materials and Production	1962	30
Study Board of Civil Engineering	1886	27
Study Board of Architecture and Design	1820	8
Study Board of Business Economics	1645	8
Study Board of Media Technology	1474	6
Study Board of Techno-Anthropology and Sustainable Design	1355	6
Study Board of Energy	1333	8
Study Board of Cross-cultural Studies	1304	5
Study Board of Medicine	1260	5
Study Board of Public Administration and Social Studies	1191	5
Study Board of Sports Science and Public Health	1166	5
Study Board of ICT and Learning	1161	4
Study Board of Chemistry, Environmental Engineering and Biotechnology	1140	39
Study Board of Health and Technology	1082	6
Study Board of International Affairs	971	2
Study Board of Sociology	963	3
Study Board of Law	930	4
Study Board of Mathematical Sciences	833	8
Study Board of Philosophy (in Danish)	747	3
Study Board of Psychology	581	4
Study Board of Education and Learning	466	3
Study Board of Art and Technology	459	1
Study Board of Social Work (Certified)	426	2
Study Board of Auditing (cand.merc.aud.)	410	1
Study Board of Business Administration (Certified) (HD)	409	6
Study Board of History	303	4
Study Board of Social Work (Master)	301	1
Study Board of English, German and Cultural Understanding	287	7
Study Board of Economics (cand.oecon)	281	4
Study Board of International Business Communication	187	10
Study Board of Danish	172	3
Study Board of Criminology	153	1
Study Board of Music	150	3
Study Board of Languages and International Studies	138	3
Study Board of Music Therapy	114	2
Study Board of e-Government (in Danish)	78	3
Study Board of Master of Public Governance (MPG) (in Danish)	49	3
Study Board of Land Surveyor Education	36	1
Study Board of Master programmes of Department of Sociology and Social Work	28	2
Study Board of Physics and Nanotechnology	0	1

11.5 Inputs

11.5.1 Identity and Access Management

Short
”Identity and Access Management”

Long
”Identity management (IdM), also known as identity and access management (IAM or IdAM), is a framework of policies and technologies for ensuring that the proper people in an enterprise have the appropriate access to technology resources. IdM systems fall under the overarching umbrellas of IT security and data management. Identity and access management systems not only identify, authenticate, and authorize individuals who will be utilizing IT resources, but also the hardware and applications employees need to access.[1][2] Identity and access management solutions have become more prevalent and critical in recent years as regulatory compliance requirements have become increasingly more rigorous and complex.[3]

It addresses the need to ensure appropriate access to resources across increasingly heterogeneous technology environments and to meet increasingly rigorous compliance requirements.[4]

The terms ”identity management” (IdM) and ”identity and access management” are used interchangeably in the area of identity access management.[5]

Identity-management systems, products, applications and platforms manage identifying and ancillary data about entities that include individuals, computer-related hardware, and software applications.

IdM covers issues such as how users gain an identity, the roles and, sometimes, the permissions that identity grants, the protection of that identity and the technologies supporting that protection (e.g., network protocols, digital certificates, passwords, etc.).”

Course
https://www.moodle.aau.dk/course/view.php?id=29489

11.5.2 Internet of Things

Short
”Internet of Things”

Long
”The Internet of things (IoT) is a system of interrelated computing devices, mechanical and digital machines provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.[1][2][3][4]

The definition of the Internet of things has evolved due to the convergence of multiple technologies, real-time analytics, machine learning, commodity sensors, and embedded systems.[1] Traditional fields of embedded systems, wireless sensor networks, control systems, automation (including home and building automation), and others all contribute to enabling the Internet of things. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the ”smart home”, including devices and appliances (such as lighting fixtures, thermostats, home security systems and cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smartphones and smart speakers.

There are a number of serious concerns about dangers in the growth of IoT, especially in the areas of privacy and security, and consequently industry and governmental moves to address these concerns have begun."

Course
https://www.moodle.aau.dk/course/view.php?id=29492