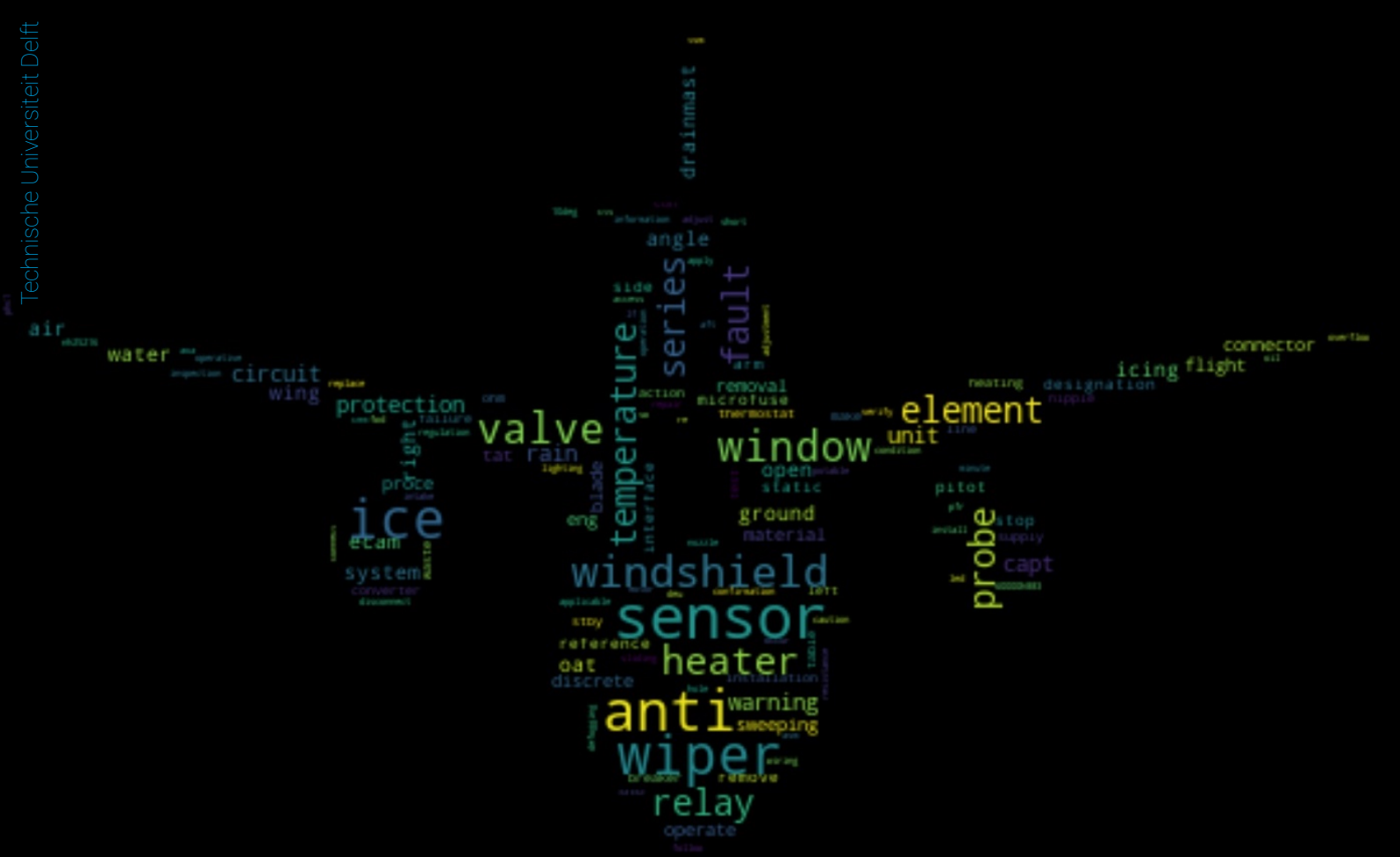


Effective use of digital aircraft maintenance documentation by implementing ranking and feedback.

G.J.H. Geurts

TU Delft

Effective use of digital aircraft maintenance documentation by implementing ranking and feedback.

by

G.J.H. Geurts

to obtain the degree of Master of Science

at the Delft University of Technology
to be defended publicly on Friday September 28, 2018 at 13:15 PM.

Student number:	4001877	
Project duration:	February 1, 2018 – September 28, 2018	
Supervisors:	Dr. Ir. W.J.C. Verhagen	TU Delft
	Ir. H. Koornneef	TU Delft
Exam committee:	Prof. Dr. R. Curran	TU Delft
	Dr. Ir. W.J.C. Verhagen	TU Delft
	Ir. H. Koornneef	TU Delft
	External Member	TU Delft

TU Delft

Preface

A challenging curriculum has to be fulfilled to obtain the degree of Master of Science in Aerospace Engineering at the University of Technology Delft. The most critical steps are to finish all courses (composed of lectures, exams, and assignments), do a challenging internship and write the final thesis. For the chosen specialization "Air Transport and Operations" the focus of the thesis is on the operational side of the aircraft business. It is my passion to streamline workflows and develop procedures to optimize the operations in the business. This thesis focusses on the maintenance domain, dealing with aircraft overhaul, repairs, and modifications. More specially, in here research is performed to answer the question if it is possible to efficiently combine a ranking method with feedback to show relevant documentation to aircraft technicians and thereby decrease their stress level. The proposed exploratory research is titled: " Effective use of digital aircraft maintenance documentation by implementing ranking and feedback".

The responsibility of the thesis lies entirely with the student from the start to the end. However, many people are directly and indirectly involved. In truth, I could not have achieved my current level of success without this strong support group. Without the help, advice, and support of all those people, this project would be incredibly lonesome. Most students mention their thesis period as one of the best in the curriculum. Unfortunately, I must admit that finishing my thesis was a bumpy road with ups but also many downs. Via this way, I would like to thank the most important people around me who supported and guided me during this time.

First and foremost, my parents, for their unconditional and continuous support during all my years in Delft. Without their motivational help, it would have been impossible even to start my MSc. Program. A special thanks to my mother, because of all her motivational speeches that made me continue my studies during the times my motivation was at an all-time low. I owe her also a sorry for the times I was not always there to help her during the times she needed me. Second, I would like to express my gratitude to my supervisors Dr. Ir. Wim Verhagen and Ir. Hemmo Koornneef. First for their belief in me to fulfill this project, but secondly also for the guidance and support. Their feedback, advice, and ideas have helped me to bring this thesis to a higher level. Thirdly, an individual thanks to my friend Tim de Boer. Without your help, and for this thesis the help in programming, in particular, it was impossible to gain these results. Also, I would like to thank you for our last couple of years together in Delft, without your enthusiasm, countless conversations, and many study sessions, those years would not have been the same! Finally, a massive thanks to my girlfriend Shanon Hendriks and my best friends Rens Pijpers and Paul Hendrix for their unconditional support and believe in me. Shanon thank you for the acceptation during the times I could not be there to support you because I had to study. Rens and Paul, thank you for all the great times and the fun we had during the countless times together. Without the fun and excitement, you all gave me; it would have been impossible for me to come this far.

Guus Geurts
Venray, September 2018

Abstract

Aircraft maintenance technicians sometimes avoid the use of maintenance manuals, especially in cases when they experience to many pressure to perform task-specific maintenance. Literature shows the available maintenance documentation system are either slow, or it is difficult to search for task-specific information. On the other hand, literature in the information retrieval domain shows many algorithms are able to retrieve information from documents effectively and efficiently. These algorithms are often compared to the "best matching 25" algorithm. This algorithm often serves as a baseline in the information retrieval domain because it delivers equally good retrieval results as new algorithms. The algorithm is renowned because of its robustness and has provided excellent results in many fields. Literature also shows the addition of pseudo-relevance feedback to the best matching 25 algorithm might result in better retrieval performance up to 25%. This thesis fills the need of an easy to search through information system with the proven algorithm and sets a baseline in the aircraft maintenance industry for the use of ranking algorithms. To the best of the author's knowledge, it is the first time a ranking algorithm in combination with pseudo-relevance feedback is applied to aircraft maintenance documentation. Next to this domain novelty the work is also applied to the actual field making use of aircraft maintenance technicians, and the results are verified using statistical tests.

The aircraft industry is currently in the transition phase from using paper-based documentation towards digital documentation. The first steps in this transition are made, but the resulting systems are either slow, lack the ability of easily finding task-specific documentation, and are often vendor specific. To make it easier for aircraft maintenance technicians to find relevant information to perform a maintenance task, a web-based model is developed. Incorporated in this model is the best matching 25 algorithm and the option to use pseudo-relevance feedback. Due to a practical experiment, the effectiveness of the algorithm with and without pseudo-relevance feedback is investigated. Seventeen aircraft maintenance technician students are given six problem cases, from which they have to find the six corresponding aircraft maintenance manuals which describe the to be performed maintenance task. The search is done by employing a query search in the model. Next to this, the aircraft maintenance technicians also have to find the same documentation using their everyday method of finding information.

The problem cases divided into three categories, where each category consists of two problems. These categories range from frequently occurring too rarely occurring maintenance tasks. This division is made to test the hypothesis if more experience leads to the faster retrieving of relevant documentation. This hypothesis is chosen because it is expected more experience leads to the use of more specific queries, making the model able to retrieve more relevant documents, and therefore finding the correct maintenance document in a shorter time span. The first research question is: does the implementation of the best matching 25 algorithm lead to more relevant search results for aircraft maintenance technicians searching for documentation, and what is the time efficiency impact due to this algorithm, compared to the current way of working? Literature showed the addition of pseudo-relevance feedback results in better retrieval performance in other domains, therefore the second research question is: does the addition of pseudo-relevance feedback to the BM25 algorithm lead to more relevant search results compared to the BM25 alone, and what is the time efficiency impact due to feedback? The time necessary for an aircraft maintenance technician to find the correct documentation and its rank are taken as parameters to evaluate the efficiency and accuracy of the model. The test group is free to search the maintenance documents for their current way of working, but when the web-based model is used they are limited to three search attempts. The queries are written down before starting the search to make a fair comparison between the algorithm and the addition of pseudo-relevance feedback. After the search all aircraft maintenance technicians have to write down the document name, the place where they found the document, and the time it took to find it. To evaluate if the addition of pseudo-relevance feedback does lead to more accurate retrieval, the place (rank) in the top 10 results will be compared. To compare the efficiency of the current way of working against the other two, the average search time is compared.

The model is described in detail using Python code, figures, examples, and continued by a step-wise verification. A sensitivity analysis shows the time to display the results from local testing is stable, while the

results from using the model online show significant fluctuations. The use of the private server, which is limited in 1 GB of RAM memory, is mainly responsible for these fluctuations in response time. Further, the sensitivity analysis showed that, the addition of pseudo-relevance feedback does not result in better retrieval performance for all queries. In fact, for only one of the four queries, the correct document is shown higher in the list of results. It is also shown that the parameters of the algorithm and the pseudo-relevance feedback have a significant impact on the search results. The sensitivity analysis was too short to conclude which parameters were the best to choose for the given aircraft maintenance manuals. The values of both methods are chosen to be close to the once suggested in literature and fixed during the tests. The results from the practical test show 87.3% of the aircraft maintenance technicians is able to retrieve the correct document by the current way of working. On average it took the aircraft maintenance technicians 203 seconds to find this document. By using the best matching 25 algorithm the search time is lowered by 72.9% to an average of 55 seconds making 89.2% of the aircraft maintenance technicians able to find the correct maintenance manual. The addition of pseudo-relevance feedback to this algorithm did not lead to an improvement of the number of retrieved documents, and was found to be equal for each individual problem compared to the best matching 25 algorithm solely. However, the average search time is decreased further to an average of 32 seconds, corresponding to an retrieval time improvement of 84.2% compared to the current way of working. Due to the small number of participants in the experiments and the occurrences of non-normally distributed data it is chosen to perform non-parametric tests to validate the results. The Wilcoxon Signed-Rank test was found to be the best option because it fulfills all requirements. The test results show for three of the four applied tests it could be concluded with a significance level of 5% the faster retrieval results are due to the addition of pseudo-relevance feedback with a medium effect size.

This research showed for the first time in the aircraft maintenance domain the use of ranking algorithms, and the addition of relevance feedback, has a significant effect on the time to find task-specific information, while maintaining the same document retrieval accuracy. For the six cases, a decrease in average search time from 203 seconds to 32 seconds was possible. It was found the hypothesis, that more knowledge leads to faster retrieval of relevant documentation, was not correct for all cases. It is expected this is due to the fact the same terms were used in the problem description as the name of the maintenance manual that was to be found. This made the algorithm able to identify this manual as highly relevant. It is recommended to test the model on the tarmac where the aircraft maintenance technicians can solve the same problem cases, without seeing the questions on paper. Future research is necessary to investigate whether other ranking algorithms or additions to the current algorithms provide better results in this domain. Also, it is required to perform additional research by evaluating the individual parameters of both the best matching 25 algorithm as the pseudo-relevance feedback code. The aforementioned methods can be combined into a research where the algorithm parameters are automatically improved each iteration by using implicit feedback in combination with machine learning techniques. As a final recommendation, more resources should be made available on efficient search methods by academia and the industry to make information better available and more accessible. The operational impact on the aircraft maintenance domain is expected to be high when the same results can be achieved by the model, in case the number of maintenance manuals is extended, and made available to the industry.

Contents

List of Figures

List of Tables

List of Abbreviations

AMM	Aircraft Maintenance Manual
AMT	Aircraft Maintenance Technician
ATA	Air Transport Association
BM25	Best Matching 25
HTML	Hypertext Markup Language
IATA	International Air Transport Association
IDF	Inverse Document Frequency
IR	Information Retrieval
MRO	Maintenance, Repair and Overhaul
PDF	Portable Document Format
SGML	Standard Generalized Markup Language
TF-IDF	Term Frequency - Inverse Document Frequency
TSM	Trouble Shooting Manual

Nomenclature

avgdl	Average document length in words
b	Free parameter for document length normalization
D	Document
f (q_i **, D)**	Term frequency of the query in document D. The number of times a query q occurs in a document D
k_1	Free parameter for term frequency scaling
N	Number of documents in collection
n (q_i)	Number of documents containing query i
Q	Query
q_i	i^{th} Query
|D|	Document length in words

1 Introduction

For years, air transport is well-known as one of the safest forms of transportation. The aviation consultancy firm To70[29] concludes in their safety review "in 2017 the fatal accident rate per flight is as low as 0.08 per million flights for large aeroplane in commercial air transport. That is a rate of one fatal accident for every 12 million flights". To be a form of transport this safe is only possible by coordinating all activities across the entire industry strictly. These activities range from complex policy changes (e.g., lower the number of landing rights of an airport) to better training facilities for personnel. Maintenance howsoever plays a crucial role, to maintain this high standard, in aircraft safety for decades. To keep an aircraft airworthy many maintenance tasks need to be performed to comply with all regulations, i.e., many aircraft parts need to be inspected or replaced when the limit is near. These tasks are known in advance because one knows upfront maintenance (or at least inspection) is necessary and therefore named "scheduled maintenance". During maintenance execution, an aircraft is not earning money for an airliner and expected from the aircraft maintenance technicians (AMTs) to perform maintenance in the shortest time possible. Unfortunately, it is not possible to plan all maintenance tasks upfront, e.g., during an inspection round a mechanic sees some engine blades are deprecated. This type of maintenance, not known in advance, is called "unscheduled maintenance". According to Suwondo [28], the ratio of unscheduled/scheduled maintenance costs for a 20-year-old aircraft grows to a factor 2 and increases with aircraft age. One reason for the increase of the expenses is unscheduled maintenance, which is often found during inspection when the aircraft is still in operation. Mechanics must perform their tasks during the available turnaround time. Turnaround time is the time an aircraft is at the gate and leaves the gate again for a new flight. The time available to perform this maintenance is limited further because it is prohibited to perform maintenance when passengers are near the aircraft. This small time frame, in which support is possible, puts enormous pressure on aircraft maintenance mechanics. According to Lampe [19], 15-30% of the total maintenance time is lost in finding relevant documentation to perform the repair safely [1]. For unscheduled maintenance the expectation is this requires even more time.

Industry, in general, is always in search of new technologies to decrease people's workload and reduce the number of errors. Especially the aerospace industry is famed for its use of high-level technology and has a long history of being an early adopter, innovator, and investigator of new technologies. With the aerospace industry on top of the number of initiatives, the term "Industry 4.0" (mentioned also as the fourth industrial revolution) is finding its way to many industries to digitalize the world, resulting in decreasing costs and increasing information accessibility [4]. With a significant percentage of revenue spent on maintenance, repair and overhaul (MRO)activities, research is still growing each year [10]. In 2014, "Globally, airlines spent $62.1 billion on MRO, representing around 9% of total operational costs [12]." Although being on top of the many Industry 4.0 initiatives, daily practice shows the use of paperless documentation for task support in line-maintenance operations is limited [31]. According to the International Civil Aviation Organization: "the implementation of the electronic aircraft maintenance records poses challenges such as electronic signature, security and integrity of records, and transferability from one record system to another" [14]. Industry offers some solutions to make use of digital maintenance documentation but users mention these applications come with several significant downsides; they are vendor specific, not designed for mobile use, slow in performance and their search functionality is insufficient [17]. These applications often consist of a central digital

[1] This statement results from research using paper-based maintenance documentation.

library, in which users can search on keywords. The system shows the results which are exact case-sensitive keyword matches. Due to the massive amount of documents, the AMT is left over with an unreasonable amount of results. Sometimes the applications offer another way of finding documents, e.g., a click-through menu which in most cases is ordered by the Air Transport Association ATA chapter. Mechanics need to know exactly where to find specific documents or they end up in an endless search. It is required to increase the efficiency of digital aircraft maintenance documentation systems. Efficiency means the AMT retrieves the most relevant document in the shortest time possible. It is worth mentioning the companies Google and Yahoo are trying to accomplish this task already for years. Both spend time and money on ranking algorithms and user-feedback to retrieve web pages from user queries. Ranking algorithms are mathematical approaches where documents or information are listed in some order depending on the algorithm variables. Their task is to order these documents so the user can find information in the shortest time possible. It is therefore remarkable these approaches are absent in the aerospace maintenance domain.

The combination of time pressure and a lack of a dedicated systems puts enormous pressure on AMTs, especially in the case of unscheduled maintenance, where he is confronted with a choice. Save some time by avoiding the use of documentation, thereby ignoring regulations and increase the (unnecessary) risk or accept the risk of flight delays by searching for task-specific documentation. Due to this impossible choice, the following defines the problem statement of this thesis:

"Aircraft maintenance technicians are not always able to find the correct maintenance documentation and perform their task during turnaround time."

This thesis attempts to investigate and set a baseline for improving access to digital aircraft maintenance documentation with a focus on simplifying the retrieval of relevant information for (unscheduled line) maintenance tasks. Data for this study is collected by several practical experiments, and the results provide new insights into the effect of using the Best Matching 25 (BM25) algorithm on technical content. Additionally, a test is performed to understand if the addition of pseudo-relevance feedback improves the results any further and can decrease the mechanics' search time. Pseudo-relevance feedback is used to improve the retrieval performance of a search without an extra interaction from the user. This study is unable to encompass the entire collection of maintenance documents and focusses on the use of the aircraft maintenance manuals (AMMs)and trouble shooting manuals (TSMs). Although the limited amount of documentation, this thesis will contribute to the domain novelty because to the best of the author's knowledge the use the well known BM25 algorithm (plus the addition of pseudo-relevance feedback) has never been applied to aircraft maintenance documentation before. Secondly, it provides a baseline for academia and industry for the possibilities to search through digital aircraft maintenance documents.

First, a brief overview of the theoretical knowledge required to understand the content is given, and supplemented with a short literature study. Chapter 3 describes the methodology of the study including its rationale, research design, and limitations. In Chapter 4 the experimental setup is explained. The problem cases of the experiment, the sampling plan and details about how information is analyzed. Next, Chapter 5 goes step by step into detail about the primary building blocks of the model, including some pseudo-code and a verification section. The chapter ends with a sensitivity analysis to investigate for the effect of changing separate parameters of the model. Next, Chapter 6 states the results of the performed test discussing the findings. The thesis ends with Chapter 7 with some concluding remarks about its contribution to the state-of-the-art and recommendations for research.

2

Literature Review

The aircraft maintenance industry is at the beginning of the transition from paper-based documentation towards the use of digital documentation. Unfortunately, the available literature is limited towards the use of maintenance information systems ([31], [20]). On the other hand, many literature is available in the information retrieval domain about the use of algorithms and models to make information better and more efficiently accessible. This chapter is therefore split into two blocks; aircraft maintenance and information retrieval. The first two sections describe aircraft maintenance in general and give state of the art in the domain. In the subsequent sections, the focus lays on the information retrieval (IR) domain with a focus on the classical models and relevance feedback. The final section combines information from the two fields, thereby showing the possibility of filling the gap in aircraft maintenance documentation retrieval by using algorithms from IR.

2.1. Aircraft Line Maintenance Operations

Unlike most base maintenance tasks, which is maintenance performed in the hangar, line maintenance is not always scheduled. During a flight, unforeseen problems can arise which requires new maintenance. Before the actual execution, it is essential mechanics are well prepared to avoid delays[16]. In Figure 2.1 a schematic overview is shown of a case of line maintenance where all tasks are known before the mechanic arrives at the aircraft. First, the maintenance information system, also known as computerized maintenance management system, is providing all tasks to perform. These tasks are taken out of the system by the lead engineer and bundled into workable packages. When done, he distributes the work packages to the AMTs. The work packages show the necessary maintenance on specific aircraft, but also include corresponding documentation and the equipment list required to perform the maintenance.

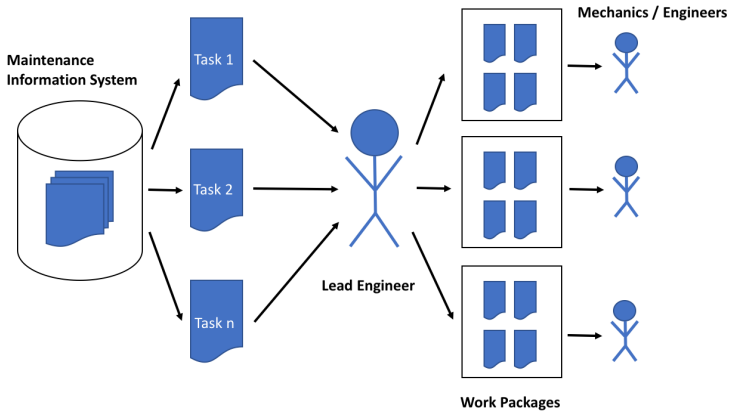


Figure 2.1: Schematic overview of work distribution from maintenance information system to mechanic.

The next step for the AMT is to prepare the maintenance by searching for the correct documentation and

equipment to perform the task. Most airliners work with a central library where mechanics can search for information. Once found, the mechanic prints the proper documentation. Before the AMT can go to the gate or stand, he has to make sure to take all equipment and parts (in case of replacements). Finally, he can take all the equipment, parts, and documentation to the gate. Once all passengers have left the plane, the AMT can start the repair.

Besides repairs, an AMT has to do an inspection round to check if any defects are present. In case defects are present, the mechanic has to decide if it is possible to defer the defect. Deferring means to delay the repair until a later moment when the aircraft is (in most cases) out of service. A mechanic always first tries to defer the defect, because of delay costs. However, in some cases, it is not possible to postpone the delay (especially when safety is affected), and the mechanic has to perform the repair. When this is the case, the AMT is faced with a difficult choice because there is a need for information at the gate, but the information can be found in the central library, which is often in the hangar. As a first option, one can call the lead engineer or another colleague to bring the required documentation to the gate but someone has to be available. A second option is to drive back to the hanger which requires time. This time, also called non-value-added time is identified by the International Air Transport Association (IATA) [13] as a potential productivity metric to be improved by the use of digital information. Both options are not ideal because it requires extra man hours or interruptions of work. Sobbe [26] et al. show interruptions have a negative effect on work flow and represent high potential for frustration. They further mention AMTs develop their own strategies for efficient information acquisition, introducing safety risk. By developing personal strategies to acquire information, the next step is to take some procedural shortcuts to perform the maintenance without consulting documentation[33].

Figure 2.2 shows a schematic overview of the previous text. In case of scheduled maintenance and only planned repairs, the process starts at the left and continues to the right. For unplanned maintenance, it is also possible to move to the left (search for tools, parts, and documentation).

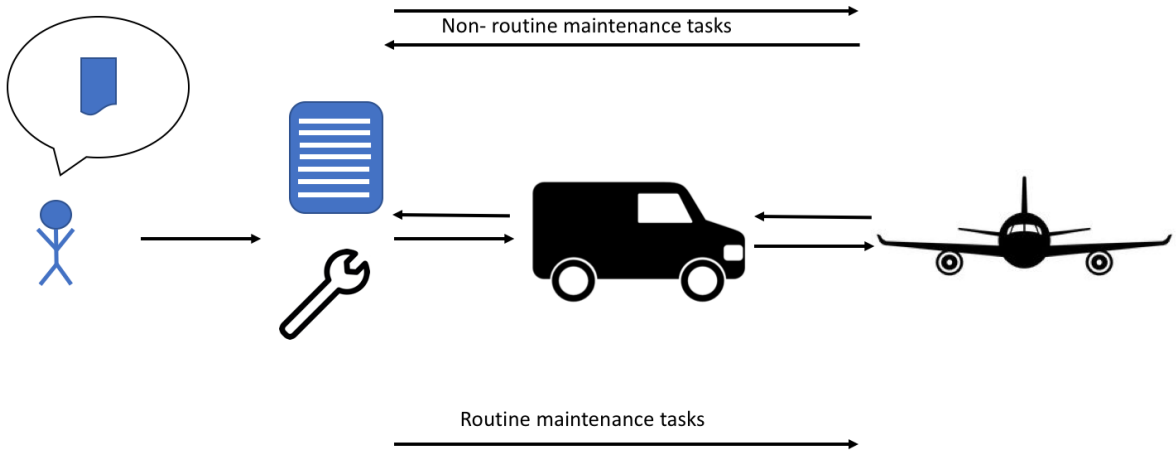


Figure 2.2: Schematic overview of line maintenance including routine and non-routine tasks.

2.2. State of the Art

Aircraft can fly for over thirty years and with some modifications it is possible to extend their lifespan even further. One should expect this long lifespan is only possible when dedicated maintenance systems are in place. Already in 2009, Candell et al. [3] pointed out document-oriented, paper-based approaches are increasingly expensive to produce and offer poor usability in MRO. However, there is a relatively small body

of literature concerned with paper-less aircraft maintenance [31], [20]. Most available literature is related to the optimization of aircraft maintenance, discussing new techniques such as augmented reality or improved maintenance planning. It is therefore IATA started the initiative and assists in the development of paper-less aircraft operations in 2017. Paper-less aircraft operations are much more than solely paper-less maintenance and Figure 2.3 provides the focus areas of paper-less operations in a roadmap.

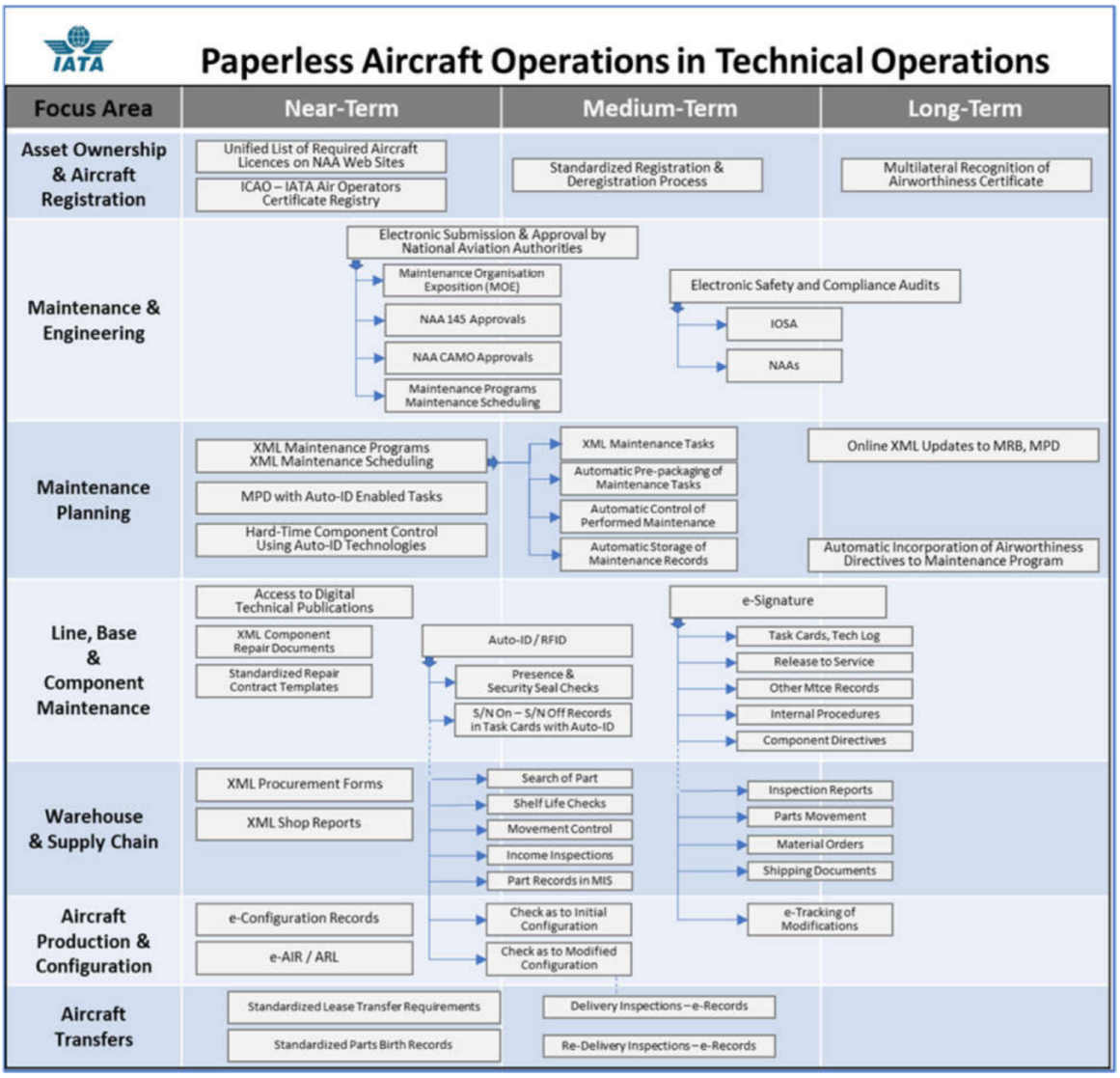


Figure 2.3: IATA roadmap towards paper-less aircraft operations, adopted from [13].

Available literature describes an approach for the use of paper-less documentation in the maintenance industry best is found by the concept of eMaintenance. eMaintenance is a broad concept including systems, techniques, methods and described by Baldwin [1] as: "E-maintenance = Excellent maintenance = Efficient maintenance (do more with fewer people and less money) + Effective maintenance (improve RAMS metrics) + Enterprise maintenance (contribute directly to enterprise performance)". Another description that fits well into the context is of Levrat et al. [21]: eMaintenance is concerned with the use and integration of IT solutions in the maintenance domain. Literature about eMaintenance starts in 1997 with the Integrated Diagnostic System [32]. This project aimed at simplifying the approach to identify and extract specific information using a rule-based approach. It turned out it was time-consuming, especially when the amount of information increased. The work of L. Scherp [25] provides a good summary of the most relevant eMaintenance projects of the last years. Starting from 2007, ontology-based approaches become popular. Verhagen and Curran [31] demonstrated the use of an ontology in aircraft maintenance as a proof of concept. In the subsequent work,

Koornneef [17] in 2015 developed a functional framework as a proof of concept for automated provisioning of contextualized maintenance documentation.

Not only does literature provides little information about digital aircraft maintenance, but also practice shows the use of aircraft maintenance documentation is still paper-based ([19], [31]). John Maggiore, director of maintenance and leasing solutions for Boeing Digital Aviation, mentioned in 2016: "There are literally millions of boxes of paper-based documents, which would circle the Earth 25 times if laid end to end. For now, the MRO industry is just in the early stages of transitioning toward electronic formats" [23].

Little academic literature regarding maintenance information systems is found, but practice shows digital solutions developed by aircraft manufacturers, airliners, and MRO companies or independent software vendors are available. Airbus is working on a tool AirN@v, Swiss has developed AMOS, and also independent software provider Ultramain Systems has an in-house developed tool. All of them provide functionalities to perform (line) maintenance but focus on scheduled maintenance where the tasks are known upfront. The problem of finding task-specific maintenance information remains for these solutions.

2.3. Information Retrieval

Many definitions regarding IR can be found but in this thesis the definition from Manning et al. [22] is applied: "Information retrieval is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)". IR is commonly used to retrieve unstructured data from an electronic text but also applies to movies, photos and other types of files. This unstructured data refers to data with a structure not easy to convert for a computer. Although IR is not the most popular research theme in the aircraft domain, in daily life one cannot be without it anymore. The most popular search engine used today as an IR service is Google, providing the user with an easy way to access unstructured data. Web search is the most popular method of IR, but also personal search options on desktops (e.g., "spotlight" on MacOS or "Windows Search" on Windows) are easily accessible. The visual layout may differ, but the working of IR models is the same; the user enters a query in the system, the system ranks the documents according to a scoring algorithm, and the top results shown to the user.

In aircraft maintenance, tasks need to be performed using maintenance manuals which are in most cases text documents, sometimes supplemented with technical drawings. The use of models to find text documents in the IR domain is called document retrieval. Figure 2.4 shows the necessary elements to perform document retrieval. First, one needs an indexer to convert a text collection into a set of indexed documents. On the other hand, the user has a particular need for information and describes this need to the system by a (search) query. The difficulty of document retrieval lies in the comparison between the queries and the indexed documents (the same argument holds true for IR). The indexer does not know the interpretation of the query while the user does not precisely know the content of the collection. The mismatch between the information need from the user and the query expression is where IR models come in place. The book "Introduction to Information Retrieval" by Manning et al. [22] provides an extensive explanation of the principles of IR, including models and algorithms developed over the years.

In the IR domain, the three most widely known classes of text retrieval models are the exact match models, vector space models and probabilistic models [30]. The boolean models are the most often used model in the first category, in which a document either matches or not. Therefore the boolean models are suitable for expert use when one has a precise understanding of the needs and the collection, but bad for most users. Due to the lack of ranking functionalities, the result from a standard boolean search is often either too few results or too many results. It requires a lot of skills to produce a manageable amount of documents to a query [22]. The second category is the vector space models, and without providing too much general information, the main disadvantage is it does not define what the values of the vector components should be. The problem of assigning appropriate values to the vector components is known as term weighting. This problem has led to the development of the probabilistic models and a central function in these probabilistic models is the allocation of terms weights, where each search term is given a weight to make it less or more important. In the past recent years, many research has been conducted to improve the term weight allocation. The BM25 retrieval function incorporates the Inverse Document Frequency (IDF) method [24] is considered the state-of-the-art of term weighting schemes during the last decades [2][7]. It is the 25th iteration of the Okapi information

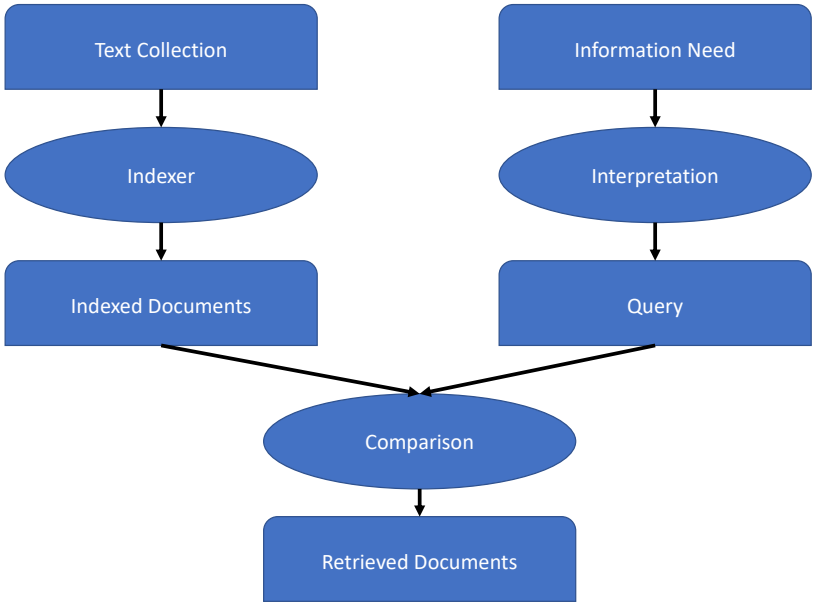


Figure 2.4: Visual summary of document retrieval, adopted from [22].

retrieval system, with the first one implemented at London's City University in the 1980s and 1990s. Not only many literature is found about the use of the BM25 algorithm, but also widely used search engine frameworks as Apache Lucene [9], and Elastissearch [5] make use of the algorithm.

No papers of applications of the BM25 algorithm in the aircraft maintenance domain were retrieved. However, outside this domain, more articles can be found to perform specific tasks. Hu et al. [11] demonstrate their Valence-Arousal-based retrieval model outperformance the BM25 algorithm to help individual efficiently locate documents and resources for their depression problems. The results show a maximum improvement of 10% in relevant documents.

An IR systems assist the user in finding information he or she needs. Typical systems do not provide an explicit answer to the user, but provides him with the existence or location of the document that might contain the needed information.The documents satisfying this information need are called relevant documents. The perfect system would retrieve only relevant documents, which in case of aircraft maintenance is only one document (although this document often refers to other documents for extra information). Unfortunately, "perfect retrieval systems do not exist and will not exist, because search statements are necessarily incomplete and relevance depends on the subjective opinion of the user"[7].

To measure the performance of an IR system literature provides two main features: efficiency measures and effectiveness measures. The first is important in terms of response time, the time between the user sending a query and receiving results back. In between this, the systems needs to: go through the index, (or sometimes build the index first) compares it with the query, ranks the documents and sends back the ordered relevant documents. As it is not the objective to improve current systems, algorithms, programming code, etc., in terms of speed no focus is given to this domain. On the other hand, effectiveness measures the relevancy of the documentation against the user query. Many general known methods are found in the literature i.e., recall, precision (at K), F-score, mean average precision but also more advanced graphical methods. All methods have a thing in common: more than one retrieved document can be relevant to a search query. Relevance can be different per user even when the same query is used. This is also the case in aircraft maintenance, but in the end, to perform a specific maintenance task it is required to have one maintenance manual describing the tasks to be performed. To evaluate a system used for finding relevant information to perform a specific task, it is preferred only this document is retrieved. As mentioned by Hiemstra [7] this is impossible and therefore the best option is to evaluate the rank of the correct document. As in reality AMTs will have

many thousands of maintenance documents available and certainly not go through all of them before starting a new search, it is wise to evaluate only a certain top n of results.

2.4. Feedback

The classical models in IR have proven to be successful but also have their limits. This section discusses a technique frequently applied as an addition to the classical models in the IR domain to improve the retrieval performance, named relevance feedback. Relevance feedback is a technique to enhance a query on the basis of relevance information. As it is unknown if any previously discussed model will retrieve relevant results, due to its specificity, it might be possible relevance feedback can offer better results.

Relevance feedback consists of three different categories, where the first is explicit supervised feedback by the user, a technique proposed already in the 1960's [6]. After a search, users can indicate the relevant documents or documents that are off-topic. The system will use this information to reformulate new queries by adding new terms, calculate a new score and rank the documents again to the modified query. The use of relevance feedback becomes unpractical for large systems and is difficult to study because the manual feedback provides an extra variable.

Many user actions can be seen as a form of implicit relevance judgements. The user can for example move documents to a folder, print documents or click on it. This information might be useful for a new search and can be stored in a log file. Implicit feedback can be a valuable source to provide the user with better information compared to the query alone. One of the difficulties is it requires time, storage space and good algorithms to make use of all information. Although users can click on specific documents, this not always implies the document is relevant. Joachims et al. [15] compared implicit feedback, due to clicks, against manual relevance judgements. They found the click-through data is informative but biased.

Both previously mentioned options of relevance feedback have shown to be useful but are impractical for larger collections of documents. Adapting the parameters in the retrieval model is time-consuming and therefore research using machine learning techniques for automatic adaption of these parameters becomes popular. However, to apply these machine learning techniques, it is required to have a large amount of training data. The aircraft maintenance domain does not have the problem of the availability of data, but the expert judgement on whether a document is relevant to a specific query or not. To avoid this problem, one can make use of the third category of relevance feedback also called: pseudo-relevance feedback. Pseudo-relevance feedback is a method for automatic local analysis to retrieve more accurate results from a search. An advantage of this technique is the independence of the user, storage space and is less time-consuming, compared to other relevance feedback methods, before results become relevant in large collection of documents. The quality of the results is strongly dependent on the top-ranked documents, because it assumes all of the top-ranked documents are relevant. The original IR system returns a list of documents where the top n documents are assumed to be relevant. From these n documents, the most relevant words are retrieved by calculating the term frequency - inverse document frequency which defines how important each word is compared to all words in the collection of documents. When the top m words are retrieved, they are added to the used query and the original retrieval method starts again.

Kraaij [18] mentions that on average, pseudo-relevance feedback improves the retrieval effectiveness by 10 to 25%, depending on the test collection, the quality of the baseline and the quality of the feedback algorithm. To prove this, he investigated the influence of pseudo-relevance feedback and its effects. He also analysed the addition of pseudo-relevance feedback to the BM25 algorithm and found the addition improved the retrieval effectiveness by 20% compared to his solution using the BM25 algorithm solely. Not only Kraaij but also Speriosu and Tashiro [27] show in their paper the addition of pseudo-relevance feedback to both a language model as the BM25 algorithm improves the retrieval effectiveness for the vast majority of queries. Important to mention is their work is performed on newspaper corpora from multiple conferences and therefore is not domain specific.

3

Methodology

The methodology of the thesis is explained in here starting with the rationale building upon the literature review, including the need for the study, the information it adds and potential utility. This section highlights the problem statement, project objective, research questions, and hypothesis. Secondly, discussed in Section 3.2 with a focus on the research design, is the scope of the project. Finally, chapter 3.3 addresses the limitations of this research and their impact.

3.1. Rationale Behind Research

Finding the correct information to perform a repair was always a time-consuming task for AMTs in no small amount of aircraft maintenance documentation. During the past years, airliners decreased the turnaround times and aircraft AMTs are often confronted with an undesirable trade-off. Either consult the documentation and thereby increase the risk of delaying a flight or do not consult it, and threatening safety. An impossible trade-off, but a result of the underlying problem, also defined as the problem statement:

"Aircraft maintenance technicians are not always able to find the correct maintenance documentation and perform their corresponding tasks during aircraft turnaround time."

As stated in the introduction, the transition from paper-based documentation towards paperless/digital documentation is still in the early stages. Digital documentation seems a promising solution to decrease search times for AMTs, but more research is necessary to know its influence in the complex aviation industry. Many players in the market, ranging from large aircraft manufactures to software vendors, search for complete digital solutions since the benefits are clear. Aircraft manufacturers can update their documentation by a simple mouse click, and thereby make the latest revisions of documents globally available within seconds. From an economic point of view, a small increase in efficiency due to software may result in substantial profit savings. However, more benefits of digital documentation will emerge when maintenance documents are easy to search through. AMTs, for example, will be able to reduce the time to look for documentation even without decreasing the actual repair time. By actually using documentation the repair becomes safer and possibly faster, resulting in a decrease of flight schedule disruptions.

Counted as the first step to make documentation digitally available, the 'paper on-screen' solution. Best-known documents in this category are the "Portable Document Format" (PDF), stored on a central server. Documentation is accessible via a computer, often placed in or near the hanger. Before AMTs go to the aircraft, they have to find and print all the necessary documents. As a result, they need to carry a lot of paper which in case of a break, or change of shift has to be thrown away and printed again[1]. In case of unscheduled maintenance, the situation becomes worse. To find the necessary documentation, AMTs first have to drive back to the hangar, find and print the documents and drive back again to the aircraft. Losing much time by driving is sometimes avoided by asking a colleague to help. This colleague needs to find the documentation, the necessary tools, and parts before bringing them to the aircraft. One can imagine this process is far from

[1] This is due to regulations as mentioned by an experienced AMT during an interview.

optimal and time is lost in many ways. Some airliners already give AMTs the possibility to access PDF documents with a smartphone or tablet. Unfortunately, these 'paper on-screen' documents are hard to search through and finding (task-) specific information is still time-consuming. A search on particular keywords may results in many hundreds, sometimes thousands, of hits. On the other hand, there is no guarantee the correct document is part of these hits because many solutions only consist of exact case-sensitive keyword matches. Literature already developed multiple ranking algorithms to deal with exact-case matches. Chapter 2 discusses these ranking algorithms, with a focus on the BM25 algorithm in particularly. The algorithm is used very frequently by researchers who develop a new algorithm, as the algorithm to defeat. However, for unclear reasons, nothing is found about the use of the BM25 algorithm in the aerospace domain. There are cases the ranking algorithm is supplemented with an additional method named pseudo-relevance feedback [2]. Since it is unknown if the BM25 algorithm and the addition of pseudo-relevance feedback have a positive effect on finding aircraft maintenance documentation, the objective of this thesis is:

"Measure the influence of both ranking and feedback, on finding relevant aircraft maintenance documentation."

To expose the effect of ranking, but also the impact of feedback, it is chosen to perform several experiments. These experiments consist of several (real-life) problem cases which describe aircraft defects and given to a group of 17 student AMTs, almost in the final year of their studies [3]. The AMTs have to find out which document is necessary to perform the repair correctly. Section 3.2 provides a more detailed description of the used ranking algorithms and feedback in the aerospace domain. The goal is not to deliver the best ranking algorithm, but to compare the often used BM25 algorithm (in combination with pseudo-relevance feedback), often used as a baseline in the IR domain, against the current way of working. Before the influence of ranking and feedback is investigated it is necessary to have a baseline of the current way of working ('paper on-screen' solution). This way of working is tested first, where the number of correct documents found by the AMTs and the total search time is part of the analysis. After setting this baseline, there is a need to discover the influence of the addition of the BM25 algorithm on the AMTs' search time and accuracy. The final step in the experiments is to solve the problems where the AMTs make use of the BM25 algorithm in combination with pseudo-relevance feedback. This results in the following two main research questions:

Does the implementation of the BM25 algorithm lead to more relevant search results, compared to the current way of working, for aircraft maintenance technicians searching for documentation, and what is the time efficiency impact due to this algorithm?

Does the addition of pseudo-relevance feedback to the BM25 algorithm lead to more relevant search results compared to the BM25 alone, and what is the time efficiency impact due to feedback?

This research implements the algorithms in an already existing web-based solution named Macster. Since it is web-based, Macster is independent of an operating system, but more importantly, it is available to all AMTs that have internet access. In the next chapter, one can find more information about Macster. For additional information one is referred to the article of Koornneef et al. "Contextualising aircraft maintenance documentation" [17].

Before explaining the precise working of the model implementation, additional author's hypotheses are stated, serving as a mean to answer the research questions. These hypotheses are:

The use of the BM25 algorithm leads to a higher percentage of correct documents found by AMTs and also significantly decreases the average search time compared to the current way of working.

The addition of pseudo-relevance feedback to the BM25 algorithm leads to a higher percentage of correct documents found by AMTs and also significantly lowers the average search time compared to the BM25 algorithm solely.

[2] Feedback in the IR domain is not similar as feedback in the aerospace industry, more details about feedback can be found in Chapter 5.1.7.

[3] From now the students are called AMTs. The study to become an AMT takes four years, at the time of testing there were two months left before the students went to their fourth year.

3.2. Research Design

Very idealistic, but the ideal system for AMTs, is one showing the correct document after one search. At the moment it is impossible to reach this goal, but the industry is asking for new systems that at least decrease the search time compared to current systems. The first step in this development is to make use of search queries and show the documents in decreasing order of importance. To know the importance of a document one can use a ranking algorithm, explained in more detail in section 5.1.6. Ranking algorithms make use of search queries that can be very specific for aircraft maintenance documents. There is no library from which one can grab technical search queries for the aircraft maintenance domain, so the simulation of different randomly selected queries is impossible. To do an experiment coming as close as possible to reality it is best to make use of AMTs knowing the jargon. To perform tests with employed AMTs during working hours is almost impossible. Either the number of AMTs available for an experiment was too low to get representative results, or the bureaucratic process took too long with respect to the research planning, or simply due to the fact the AMT's time was too valuable for conducting experiments. After some contact with "Techniekcollege Zuid-Limburg", the possibility was given to do tests with a group of third-year aircraft AMT students. Although these students do not have as much experience as AMTs, it is now possible to test with a larger target group and to have more time available for execution of the test. This is considered the best option given the timeframe of the project.

Scope

As time and resources are limited, first the scope is defined. Every item is mentioned in bold together with a short explanation why it was necessary.

- **Maintenance documentation from Airbus for the Airbus A318, A319, A320 and A321 family:**
  During the development of Macster solely maintenance documentation of the Airbus A318 - A321 family is used. More specifically, the AMMs are implemented which will be used during testing. Also the option to make use of the TSMs is available. The TSMs will only be used in the sensitivity analysis and not during the test because "Techniek College Zuid-Limburg" is not in the possession of these manuals. It is chosen to stick to these two types of documents, to avoid particular aircraft manufacturers implementation problems. The working of the algorithms is not affected. Also, one can see the addition of extra maintenance manuals as a possible addition for future versions of Macster.

- **English documentation:**
  Dutch AMTs will carry out the tests and it is expected all of them can read and write English. However, not expected is that all of them might be able to read and write other languages. As it is not the objective to deliver the best working algorithm, the implementation is limited to English documents. Implementing documentation from a different language does not have an added value to this research. Also, the majority of the MRO and manufacturer market is English-based.

- **Maintenance manuals:**
  To set a baseline for the current way of working, the documents available in Macster should also be available at the test location in PDF format. Found during the development of the problem cases is that at the test location the number of Airbus documents was limited. One of the few complete and available chapter was number 30 of the AMM. After review of an experienced aircraft AMT, the decision is made to use this specific chapter. The possibility to search through the TSMs is implemented but not used during the experiment. The algorithms will not make use of the TSM documents during the calculations. The reason to implement the TSM is to extend the number of documents during the sensitivity analysis. Limiting the collection to solely AMMs, has its effect on the ranking calculations. Chapter 5.2 discusses this impact during the sensitivity analysis.

- **Model:** In aircraft maintenance laptops and tablets are already used to a certain level. Therefore it must work at least on these devices.

- **Technical drawings:** It is decided to exclude technical drawings. The functionality to show technical drawings is not available in Macster and left as a recommendation for future versions. Also, different ranking algorithms are required to perform ranking of (technical) drawings compared to the ones used for document retrieval [22].

3.3. Limitations

The research is bounded by its scope and assumptions due to a limited amount of time and resources. This has its effect and an overview of the main limitations can be found below. Each limitation is stated in bold, followed by an explanation of why it could not be avoided and the impact it has on time and accuracy.

- **Number of ATA chapters used:** Only ATA chapter 30 of the AMM is included in Macster. The reason to choose for only one chapter is simple: limit the research. After an interview with two experienced AMTs, it can be assumed AMTs have enough knowledge to at least know the ATA chapter number in which the documents related to a problem can be found after a short period of time. The reason to choose for chapter 30, in particular, is because during the tests it is necessary to have a specific chapter available in Macster, but also as a PDF document.
  Impact on search time: By limiting the number of ATA chapters, the number of documents is also limited. At the moment it is expected this is positive for the calculation speed because the server is capable of performing complex calculations.
  Impact on accuracy: In case the knowledge of the AMTs is enough to immediately select the correct chapter, there is no impact on the accuracy. However, if a general search is started, it depends on the specificity of the queries if there is an impact on accuracy. When the queries are specific enough, the top 10 results will not be affected. However, when queries are used which can be found in every document it is expected to have a large impact on accuracy.

- **Type of aircraft:** The used aircraft fleet is the Airbus A318-A321 family because the documents of this family were already implemented in Macster.
  Impact on search time: The impact on search time for this research can be neglected, but one must be careful when a full implementation of Macster at a maintenance service provider would be planned. The impact can range from very low when only Airbus aircraft are maintained to very high when aircraft come from multiple manufacturers. In this case, an extra option to filter on aircraft type would be a solution.
  Impact on accuracy: The same explanation holds as for the impact on search time.

- **Students:** The group participating in the experiment consists of students. Therefore their knowledge is lower compared to AMTs already working in the industry. Although their knowledge is less, it is considered the best option for the experiment because it is extremely difficult to get AMTs available for testing purposes.
  Impact on search time: The impact on search time is low because students using a laptop or tablet might compensate for the possible extra time spent on searching due to inaccurate queries. Students might use different queries but the use of different queries still needs to be investigated, so it is impossible to draw any conclusions on this effect.
  Impact on accuracy: The expectation is that more specific queries will lead to better results. As students have less experience it can be expected the detail level of the queries is slightly lower which might affect accuracy in a negative way.

- **Server with 1 core of 1GB:** Macster was already installed on this server and by the start of the project a much faster server was delayed. As an end solution, it is a must to have a better server because it is not possible to make use of Macster with over ten people.
  Impact on search time: The search time is certainly affected because a server with more memory is able to perform the necessary calculation faster. For every iteration, this is estimated as a couple of seconds. Although this seems to have a big influence, the expectation is that this time is negligible compared to the time it takes AMTs to decide whether a document is correct or not. In the end, one can do a sensitivity test with the new server and make a correction for the time it took to perform the calculations. Further the impact is limited by splitting the group of AMTs during the experiments. The server therefore does not have to perform all the calculations at the same time.
  Impact on accuracy: Accuracy is not influenced, because the same calculations are performed.

- **Number of participants:** The test will be performed with 17 AMT students, a small number of test persons for an experiment. Making conclusions about the separate systems, the power of these conclusions is low. It is, however, better to have 17 AMTs during a test, as to perform a test with less than five experienced AMTs. The chosen group is almost in their final year and had multiple internships,

therefore their experience is sufficient enough to be valuable for testing.
Impact on search time: The search time is not influenced.
Impact on accuracy: The accuracy is not influenced.

- **Tracking of time:** The time to find the correct documentation is manually tracked.
  Impact on search time: As long as the AMT stops the stopwatch immediately after he finds the correct document there is no impact. The impact can be significant in case the timer will be forgotten. Before the experiment it will be told to the AMTs to not forget the timer. In case they do forget, it is asked to mention this so the results can be removed.
  Impact on accuracy: Accuracy is not influenced.

4

Experimental Setup

This chapter describes the details about the test setup. The first section discusses the problem cases, which are developed in collaboration with an experienced AMT. Section 4.2 provides more insight in the workflow the AMTs have to follow for each method. In here also the question- and answer sheet, which is provided to the AMTs, is discussed. The results from the test will be evaluated by the method described in Section 4.3.

4.1. Problem cases

The idea behind performing experiments using AMTs is to come as close as possible to reality. Several methods exist to evaluate the performance of ranking and feedback algorithms. However, due to its specificity, it is highly essential to use queries from the jargon. Something which is only possible by performing the experiments with people who work in the field. In reality, AMTs face many thousands of problems and many different repairs. Their knowledge is broad but sometimes also specific and therefore it is impossible to develop problem cases as a layman. In Table 4.1 the problem cases developed in conjunction with an AMT with over twenty years of experience are shown. The hypothesis is that the use of specific queries is dependent on the knowledge of the AMT. Performing a task on almost a daily basis, AMTs gain more knowledge about the content of the maintenance document. The AMT can search for specific and detailed queries because more is known about the equipment, parts, error codes, etc. Expected is that using these more specific queries, results in finding the correct document high at the top of results. The opposite holds, for tasks performed very rarely because an AMT does not know much about the content or the equipment to use. It is expected this results in the correct document not shown in the top of retrieved documents when general queries are used.

Three different categories of problems are defined to test if the hypotheses are correct. For every category two problems (A and B) are developed, this to lower the possibility of finding the right document by chance. The first category defines two problems AMTs often come across. The second category is one with regularly occurring problems. Rarely occurring problems can be found in the third and final category, meaning the mechanic did not perform the task before. Table 4.1 provides a summary of the developed problems. The problem number in the first column defines the category. The second column contains a description of the actual problem given to the mechanic. The to be found document is shown in the third column.

Problem number	Description given to mechanic	To be found document
1A	The Captain of flight TP436 mentions to you that on his side the fluid that during heavy rain is sprayed onto the windshield is empty. This fluid normally prevents water droplets from adhering to the windshield outer surface and thereby increases visibility.	AMM-30-45-00-600-002-A: Servicing of the rain repellent system.
1B	The flight compartment windows are electrically heated. The windshield panels are heated for ice protection and defogging. The two side windows (sliding and fixed) are heated only for defogging. It is asked to you to perform an operational test on this system.	AMM-30-42-00-710-001-A: Operational test of the windshield anti-icing and defogging.
2A	An aircraft is in operation, and you are given a task card that describes you have to perform a test on the system responsible for the protection of the engine during icing condition on ground or in flight.	AMM-30-21-00-710-002-A: Perform a test of the operational engine air intake ice protection.
2B	An operational test of the anti-ice valve revealed intermitted operation. It is suspected this filter needs to be removed.	AMM-30-11-51-000-002-A: Removal of the anti-ice valve filter.
3A	During functional testing, it was found the left-hand windshield wiper is inoperative. You have one hour left to perform a repair. You have to make sure this system is deactivated to avoid more defects.	AMM-30-45-00-040-003-A: Deactivation of the windshield wiper.
3B	A steward complains there is a problem with the potable water. In the electronics compartment, where the potable water supply pipelines are shrouded to prevent water spillage, an inner line heater is used to prevent the water from freezing. You have to remove this system.	AMM-30-73-54-400-001-A: Removal of the inner line heater.

Table 4.1: Problem cases developed in collaboration with AMT expert and corresponding document to find.

Macster

Previous research on the transition from paper-based maintenance documents towards digital documentation resulted in the development of Macster. The system can transform the original maintenance data from Airbus, written in the Standard Generalized Markup Language (SGML), into the common used Hypertext Markup Language (HTML). The HTML structure is used for 25 years and is the building block for today's websites. Macster is a web-based system with the advantage of being independent of the type of operating system and can be accessed by every device connected to the internet.

For this thesis, it is important to mention the use of Macster is only as a visual framework. Macster can show the content of the maintenance documents but is not able to search through these documents. This work adds the possibility, and functionality to Macster, to search for task-specific information using search queries.

4.2. Sampling Plan

The tests are developed to clarify if the ranking algorithm and the addition of feedback lead to better search results or if they can reduce the search time. As stated before, it is essential to have a baseline for the current way of working, which is by using PDF documents. Before the test, all 17 AMTs will receive a question- and answer sheet together with an overview of all problem cases. First, the AMT starts reading the problem and the questions asked. The AMT is asked to find the task-specific document that provides a working method corresponding to the indicated problem. When the problem is evident, and the AMT is ready to start his

search, he can start the timer (stopwatch). The mechanics have full freedom to do their search, meaning no method is instructed to them as long as they stay in the PDF document. In this document, they can search by just scrolling through all pages or perform a keyword search. Immediately when the AMT finds the (according to him) correct document, the timer must be stopped. As a next step, the AMT can answer the questions on the answer sheet. Figure 4.1 shows an example of this question and answer sheet. As a final step for the search using PDF documents, the following questions need to be answered: "Name of document you think is correct (full name) + page number" and "Time it took to finish your search". The reason to ask for the page number explicitly is because every chapter of the AMM starts with a table of contents. In here the document name is shown but lacks a task description. In principle, there is no limit on the time to find the document because chapter 30 of the AMM contains all material necessary to perform the search. Figure 4.2a visually shows the working procedure of finding PDF documents.

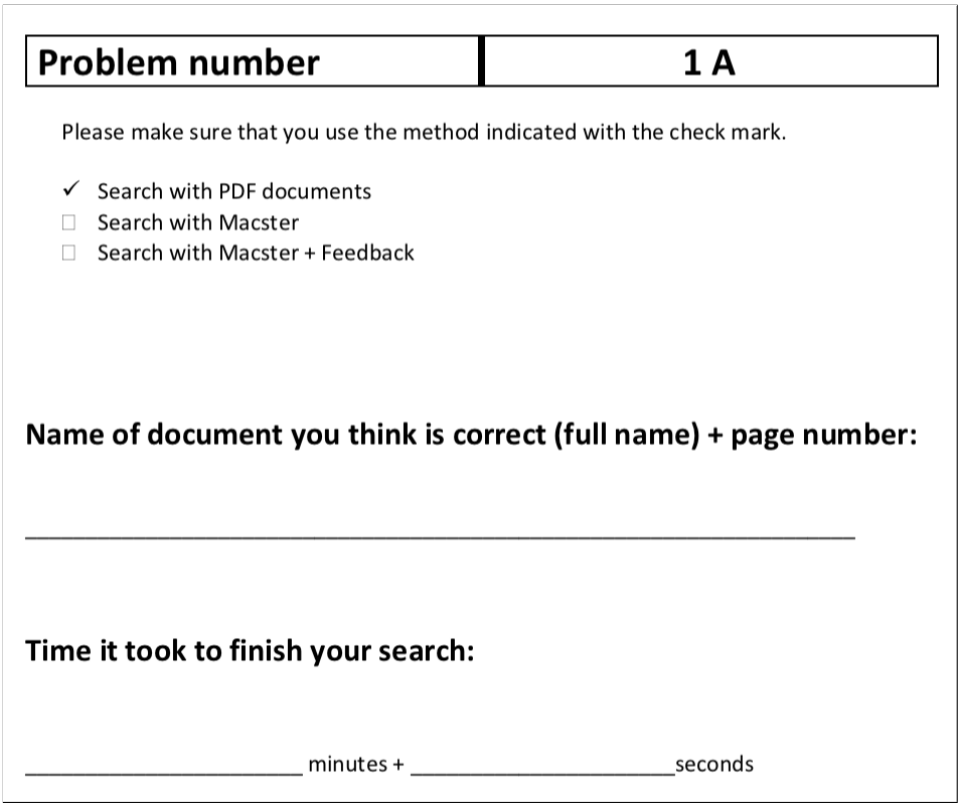
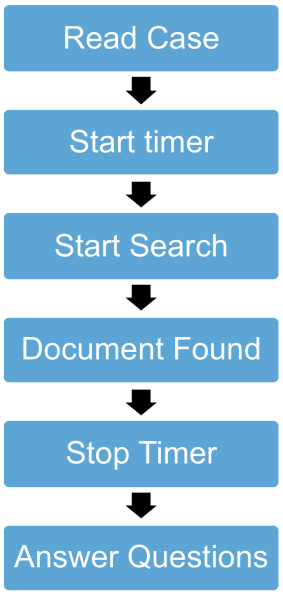


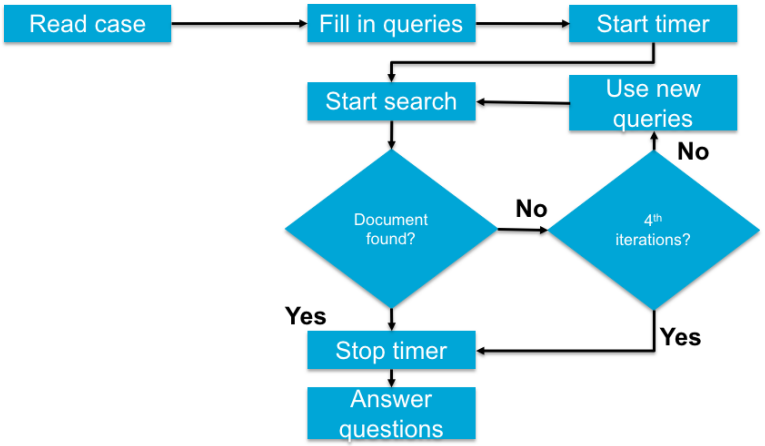
Figure 4.1: Question- and answer sheet for the PDF documents.

The working procedure for Macster also starts with reading the problem case. After reading the problem, the AMT first has to write down the queries he thinks will lead to the correct document. The mechanic has to write down three different set of queries on a new question- and answer sheet, shown in Figure 4.3. The reason for three different sets of queries is because it is possible that after the first search the correct document is not found. Required is to use the same queries are the same for both methods (BM25 with and without feedback) to make a fair comparison. Once all queries are filled in, one can copy them towards the question- and answer sheet for problems solved with feedback enabled (the content of this sheet is the same as in Figure 4.3). The next step is to start the timer and perform the search, by typing in the first set of queries on a laptop. In case the correct document is in the top 10 of results, the mechanic can stop his search and stop the timer. If not, one must use the second set of queries and can start the search again. This process repeats one more time when the AMT did not found the correct document. After three unsuccessful iterations, the AMT can stop his search and must write down he was unable to find the right document. Finally, he can answer the questions, which differ from the PDF questions. Figure 4.3 shows the questions related to the accuracy of the algorithm and the time it took to finish.

Due to the limited server capacity, it is not possible to perform the test with all AMTs at the same time. The server is not capable of performing all the ranking and feedback calculations for this amount of people.



(a) Flowchart of the work procedure for problems solved with the PDF documents.



(b) Flowchart of problem cases needed to be solved with a query search (with and without feedback).

Figure 4.2: Flowcharts of how the problems cases need to be solved, separated by method.

A small stress-test showed a drop in server performance when over ten people are performing calculations. The group is split to avoid the server performance influences the test results. During two rounds, the two groups (consisting of eight and nine persons) will perform all tests. The reason to split the group of seventeen AMTs is because one part can solve all problem cases using PDF documents, which are stored on the personal laptops and therefore not influencing the server performance. During the first round, the group consisting of eight AMTs starts by solving all six problems using the PDF document. The other nine AMTs, first solve the six problems using the BM25 method. When resolved all, the process repeats but now by using the feedback method. The second round is the same except the switching of the groups. After the second round, every AMT has solved the six problems using the three different methods. As mentioned, the AMTs make use of their laptop during the experiment [1]. Although all laptops have different specifications, the server performs all calculations. It is only the server that needs much computing power and therefore the effect on the total search time due to different laptops can be neglected.

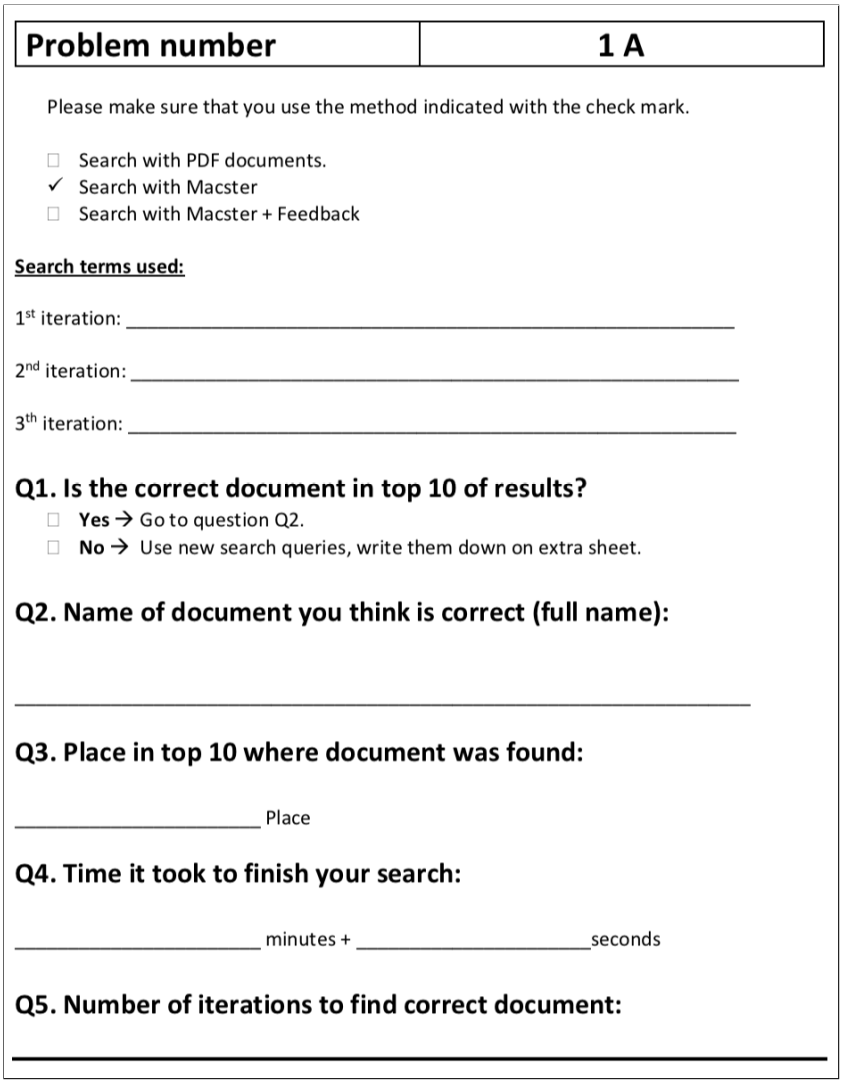
4.3. Data Analysis

As described in section 4.2 the results from the test are written down on the question- and answer sheets by the mechanics. After the test, the data is put into Microsoft Excel 2016 in order to start the data analysis.

Evaluation of accuracy

The accuracy of the three methods is analysed by the number and percentage of correctly found documents. To perform this calculation it is necessary to check first whether the AMT actually did find the correct document. Next, for each separate question, the number of AMTs who were able to find the correct document is calculated including the corresponding percentage. Indirectly, the number of AMTs who did not found the correct document, or any document at all, is found. The second parameter to evaluate accuracy is the place in the top 10 where the document was found. Only documents found in the top 10 of results (else a new iteration should be started) are asked. Also, the number of iterations to find the correct document will be taken into account, which is considered as the third method to evaluate accuracy. This is important because the development of a system which requires three iterations to find the correct document is not preferred.

[1] The use of tablets was preferred, however, due to the limited amount of resources this was not possible.



Problem number	1 A

Please make sure that you use the method indicated with the check mark.

- ☐ Search with PDF documents.
- ✓ Search with Macster
- ☐ Search with Macster + Feedback

Search terms used:

1st iteration: _____

2nd iteration: _____

3th iteration: _____

Q1. Is the correct document in top 10 of results?

- ☐ **Yes** → Go to question Q2.
- ☐ **No** → Use new search queries, write them down on extra sheet.

Q2. Name of document you think is correct (full name):

_____

Q3. Place in top 10 where document was found:

_____ Place

Q4. Time it took to finish your search:

_____ minutes + _____seconds

Q5. Number of iterations to find correct document:

Figure 4.3: Question- and answer sheet for the problems solved with Macster.

Evaluation of efficiency

The first parameter to evaluate efficiency is the time it will take the participants to find the (correct) document. As described in the sampling plan this will be tracked with a stopwatch. Efficiency is closely related to accuracy because it is very likely when the correct document is found high in the top of results, the mechanic spends less time on searching compared to the situation where he has to go through a lot of documents.

Special note of attention

Special attention must be given to the fact the test group consists of seventeen AMTs because it has an impact on the allowed type of statistic test. Since the group is small, the choice is made to perform every test on every single AMT without changing the paramaters of the model in between the tests. This type of testing is named a repeated-measures design and becomes important for the type of test to select later. Benefits of the repeated-measures design are the possibility to detect statistical differences with a smaller number of subjects but it also reduces the effects of variability because the same subjects (in this case AMTs) are used throughout the experiment [8]. Further, it is important all persons do the tests on the same day, to make sure they will not look for additional information. Finally, no changes must be made to the model in between the experiments.

It is required to do some additional statistical tests before it is possible to draw general conclusions for the results of the different methods [8]. Since use is made of only one group (no control group or extra group of AMTs perform the test), the options are limited. To compare the different methods, a before and after situation is measured (before applying BM25 and after for the comparison with PDF documents and before and after applying feedback to compare with the BM25 algorithm). Depending on if the data is normally distributed there can be decided to use a paired t-test (parametric statistical test) or the non-parametric variant the Wilcoxon signed-rank test. It is important to do this check because with a sample size of 17 (number of AMTs per test), it is not allowed to simply assume normally distributed data (sample size < 30). There are multiple options to check whether the data from the experiment is normally distributed or not. To do this, it is chosen to first do a graphical normality check by means of plotting a histogram of the data. In case the histogram is bell-shaped (peak in the middle) and is roughly symmetrical about the mean the next step is to plot a normal probability plot of the data. In case this plot is linear, it can be assumed the normal distribution is a good model for the data. In case of doubts, an extra check to look if the data follows a normally distributed pattern will be performed, namely the Shapiro-Wilk normality test. When the normality check is completed the decision is made to perform a paired t-test or the Wilcoxon signed-rank test.

5

Implementation and Verification

This chapter goes into detail about the implementation model. Individually discussed are the separate items of the main building blocks of the model, assisted with the use of Python-code and several examples. The subsequent section verifies the elements of the model. Continued by Section 5.3 is an extensive sensitivity analysis to show the effect of changing (individual) parameters in the model.

5.1. System Development

As already mentioned; Macster is the visual framework of the model, but is not able to perform a query search. Another script must be ran in the backend of the server to perform the ranking of documents. This Python script must be able to go through the documents and delivers the most relevant to the user. Figure 5.1 shows a general outline of the Python script and Appendix A includes the full Python code. Every section in this chapter highlights an individual part of the model. Starting with more information about the documents and continued by how Python can run through these documents using crawling. Users of the model can type search queries, and the model retrieves the most relevant documents related to these search queries. An inverted-index is made, such that the program only has to find the related queries in a list instead of running through all documents for every single search, described in section 5.1.5. As a next step, it is necessary to rank all documents compared to the search queries, described in section 5.1.6. In case feedback is required the pseudo-relevance feedback algorithm will be enabled before the best-ranked documents return to the user, explained in the last two sections.

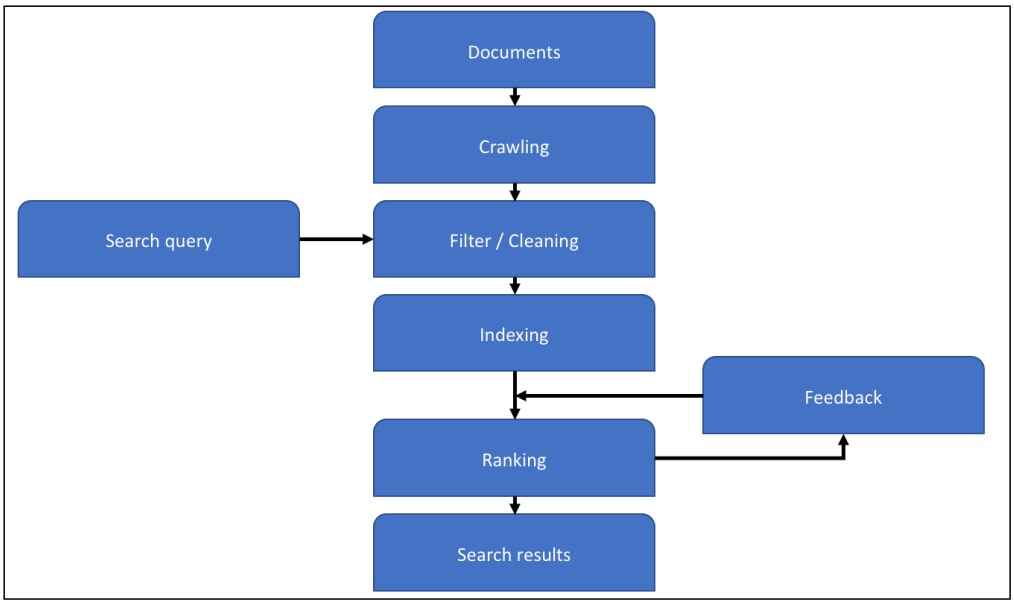


Figure 5.1: Main building blocks of the Python script.

5.1.1. Search query

One of the requirements is to make it easier to find task-specific documentation. Therefore, users must be able to type in their search queries in Macster. They can do this in the so-called "search box", shown in Figure 5.2. Users must be able to specifically look for one type of document, e.g., only AMMs or TSMs, depending on the task they have to perform. During the experiment the TSMs will not be available in Macster, but the option is present to perform the sensitivity analysis. By introducing a filter button, the specific document type selection is now available to the user. When a particular manual is enabled, this button is colored blue. Already visually implemented, but not used in this thesis, is the option to filter on ATA chapter or ATA section. The reason for this implementation is because AMTs with many years of experience often know the precise section where to find a maintenance document. Also implemented is a slider to give the user the possibility to turn the feedback algorithm on and off. In the background, an extra step must be performed in the Python script to make sure a document receives a score with and without feedback. After this thesis, one can make a final decision to make use of (pseudo-)relevance feedback or not. The on-off slider can be removed in a final version of Macster depending on the results (which option is the best retrieval method). In Figure 5.3 the newly designed slider is shown.

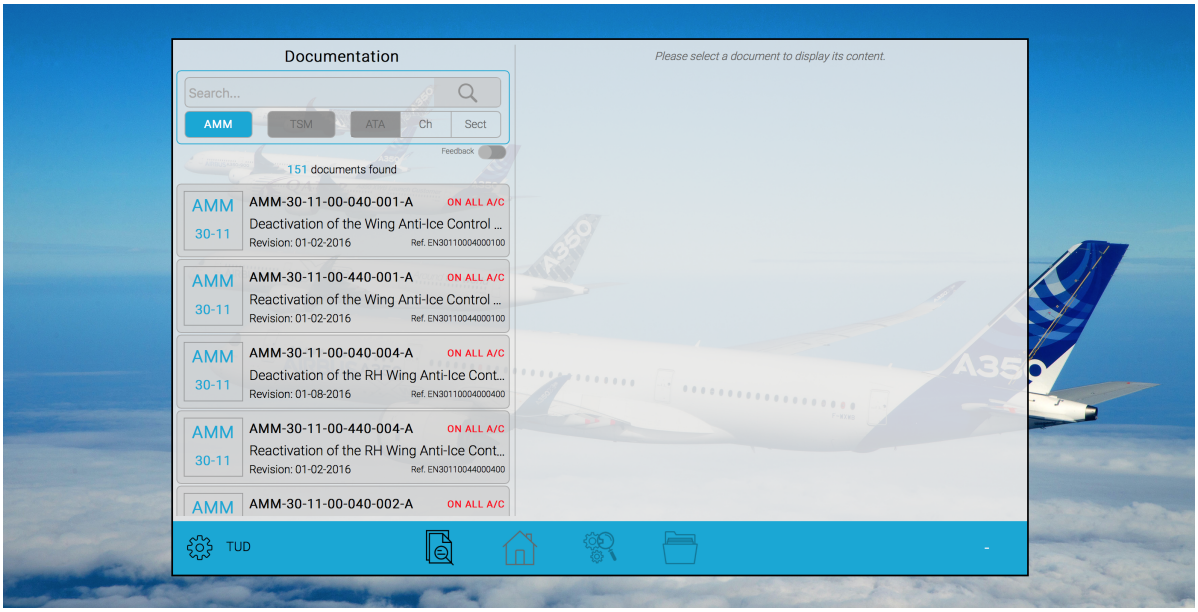


Figure 5.2: Search box in Macster.

After the implementation of the new search box, users can start their search with queries. These queries need to be copied from Macster towards the Python script because this one performs all calculations. It is possible to search on multiple queries, by typing a space between all words. These separated words are then stored into a list in Python, separated by commas and put between quotes. An example of how the query is taken over in Python: ['This', 'is', 'an', 'example', 'of', 'how', 'queries', 'are', 'separated', 'in', 'Python']. From here the exact same queries are copied into Python, including capitals and punctuation.

5.1.2. Documents

Macster makes use of HTML maintenance documents originating from the SGML maintenance documents of Airbus. The stored documentation on the web server is ATA Chapter 30 of the AMMs and TSMs (only for the sensitivity analysis) of the Airbus A318, A319, A320 and A321 family in HTML format. Due to signing a non-disclose agreement, it is not possible to show the actual AMM and TSM content.

5.1.3. Crawling

Crawling means systematically going through all documents to create an index of data. Use is made of the BeautifulSoup library in Python, to crawl through all documents in Macster. This because all documents contain many HTML tags and manually delete or replace them is impossible. Licensed by the Python Software

Foundation, and the Massachusetts Institute of Technology License 4+, the BeautifulSoup library is used all over the world. Programming time is saved, because the library is very efficient in pulling out data from HTML (and eXtensibile Markup Language) files because it recognizes most tags. The BeautifulSoup library makes it possible to separate most of the HTML tags from words. An extra implemented filter function separates the remaining tags, with an explanation in the next section. Python is now able to go through all documents and loop over the individual words. The Python code to perform this crawling is shown in listing 5.1 This listing shows for each file (AMM), it is is added to the path and read by Python. The next step is to let all files go through the filter and read them by the BeautifulSoup library. The library continues to split the words from the tags, which are again stored in Python.

Listing 5.1: Python code to crawl through documents.

```
directory = '/path/where/files/can/be/found/'
importFiles(directory):

for file in files:
        path = os.path.join(directory, file)
        document = open(path, "r")
        soup = BeautifulSoup(filter(document.read()), 'lxml')
        words = soup.text.split()
```

5.1.4. Filter

A filter module is implemented to make the queries and text both free from HTML tags, punctuation and lowercases them. This filter function is both applied to the queries and the rest of the text of the manuals, to make sure the same rules to change words are applied. HTML tags are necessary for web browsers to interpret the content in the text, for example: <table> is an HTML opening tag for a table and </table> the closing tag. The web browser interprets, all content between those tags, as part of a table. These tags are important for web browsers but do not add value for the user (therefore they are not even shown to the user) neither to the ranking algorithm.

Sometimes two words, or a word and punctuation, are bonded together. Python identifies these combinations as unique, especially at the end of a sentence because it ends with a punctuation mark. In this case problems can be exptected when the script wants to calculate a document score of a specific query. The addition of an extra line of code to the script prohibits the false classification of individual words. This code states: **re.sub(r'[-!;><?.,]$', ")** and replaces special characters by a space when they are the last character of a word.

As mentioned, the BeautifulSoup library recognizes most of the HTML tags but is not perfect. In some cases, it does not acknowledge the existence of a tag bonded against a word. This was found during the implementation of a function searching for words which is part of another word. The main idea behind this module was to implement a query suggestion module in the future, i.e., a module able to suggest queries the user possibly wants to use. After a couple of tests, it turned out this module was able to recognize words part of other words, e.g., "or" is part of "before". However, also found is the BeautifulSoup package did not function perfectly because words with HTML tags before or after were left, e.g., wing . Because of this, it was necessary to start an extensive search for words with tags against them. The tags not filtered out correctly were found, replaced by the same tag, including an extra space before and after the tag. Due to the necessity of cutting tags apart from words, all words found in the documents go directly into this filter function.

Listing 5.2: Python code to filter queries.

```
def filter(string):
    result = string\
        .lower()\
        .replace("(", "␣(␣")\
        .replace(")", "␣)␣")\
        .replace(",", "␣")\
        .replace(".", "␣")\
```

```
        .replace(":", " ")\
        .replace("</div>", " </div> ")\
        .replace("</span>", " </span ")\
        .replace("</thead>", " </thead> ")\
        .replace("</table>", " </table> ")\
        .replace("table>", "table")\
        .replace("</td>", " </td> ")\
        .replace("</th>", " </th> ")\
        .replace("</tr>", " </tr> ")\
        .replace("/ctl", " /ctl ")\
        .replace("c/b-", " c/b- ")\
        .replace("<a>", " <a> ")\
        .replace("<b>", " <b> ")\
        .replace("<i>", " <i> ")\
        .replace("p/bsw-", " p/bsw- ")\
        .replace("bsw-", "bsw")\
        .replace("oat=", "oat") \
        .replace('"', ' ')\
        .replace("'\u25b8'", " ")\
        .replace("fonctional", "functional")\
        .replace("/", " / ")
    result = re.sub(r'[-!;><?.,]$', '', result)
    return result
```

5.1.5. Indexing

Now the words are correctly separated, the second part of the crawling (create an index of data) can start. To do this, three different empty lists are created. The script goes through all documents and for each manual separately, run through all words. In case a word is identified as a new word, the script places it into the list together with the value 1. If the word is found earlier (i.e. word is already in the list), its value is increased by one. This process repeats for all files and another directory, named inverted index, is filled with unique words, the name of the document (where the script found the word), and the number of times the word occurs in this document. Finally, the total number of words specified per document is counted and stored into a list with the file name.

Listing 5.3: Python code to index words from documents.

```
def importFiles(directory):
    invindex = []
    totalUniqueWords = defaultdict(lambda: 0)
    totalWordsPerFile = {}

    files = os.listdir(directory)
    total_num_words_all_files = 0
    for file in files:
        uniquewords = Counter(words).keys()
        values = Counter(words).values()
        for uniqueword in uniquewords:
            if uniqueword != totalUniqueWords:
                totalUniqueWords[uniqueword] += 1

        for i in xrange(0, len(uniquewords)):
            invindex.append([uniquewords[i], file, values[i]])

        totalWordsPerFile[file] = len(words)
        total_num_words_all_files += len(words)
```

```
def numberOfDocumentsPerTotalUniqueWords(files, numberOfDocuments,
                                         invindex, totaluniquewords):
    allWords = []
    result = {}
    for inv in sorted(invindex):
        if inv[1] in files:
            allWords.append(inv[0])

    for word in Counter(allWords).keys():
        result[word] = 0
        for inv in invindex:
            if inv[0] == word:
                result[word] = result[word] + 1
    return result
```

5.1.6. Ranking

The ranking algorithm to give every document a score related to one or more specific queries is the BM25 algorithm, with its mathematical formulation in Equation 5.1. A document D does not have a static score, changes for every query Q from search to search as indicated by $BM25_{Score}(D,Q)$.

$$BM25_{Score}(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \tag{5.1}$$

with:

$$IDF(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \tag{5.2}$$

With:
IDF: Inverse document frequency
N: Total number of documents in the collection
n (q_i): Number of documents containing query
q_i: i^{th} Query
Q: Query
D: Document
f (q_i , D): Term frequency of query Q in document D
k_1: free parameter
b: free parameter
|D|: Document length in words
avgdl: Average document length

An important part of the BM25 algorithm is the calculation of the IDF shown in Equation 5.2. The IDF relates the importance of a single word to all words in the collection. Frequently occurring words, e.g., ”the”, are automatically filtered out by the equation, while infrequently used words receive a high score. An example best explains the equation. For this example, one takes a collection of 100 documents (N = 100). Now the importance of the two words ”the” and ”wing” is compared. The word ”the” is found in 97 documents and the word ”wing” in 10, so n (q_i) = n(”test”) = 97 and n (q_i) = n(”wing”) = 10, resulting in equations 5.3 and 5.4. The documents are called top-ranked documents when they have received a high score. Words occurring only in a few documents, therefore, receive a higher IDF score.

$$IDF(''the'') = \log \frac{100 - 97 + 0.5}{97 + 0.5} = \log \frac{3.5}{97.5} = -1.44 \tag{5.3}$$

$$IDF(''wing'') = \log \frac{100 - 10 + 0.5}{10 + 0.5} = \log \frac{90.5}{10.5} = 0.94 \tag{5.4}$$

Interesting cases are the extremes; when words (e.g., "the") are found in every manual or found only once in a single document while the number of documents is high. Equation 5.5 shows for the first case the limit goes to $-\infty$, meaning this word punishes all documents extremely hard. The second case represents a real-life scenario when an AMT is in search for a document, does not know where to find it and starts a general search in all maintenance documents. For this case, shown in Equation 5.6 it is found the limit goes to ∞, meaning when this word is used as a query, this specific document will receive a massive score and therefore always shown first. Luckily this is only in extreme cases because the IDF values for finding a word in every document and finding the word once, for $100 \cdot 10^6$ are -8.301 and 8.301. The expectation is this will never occur because when a maintenance provider incorporates all of its maintenance documents it will not exceed the amount of $100 \cdot 10^6$ documents.

$$\lim_{n(q_i)\to\infty} \log_{10} \frac{0.5}{n(q_i) + 0.5} = -\infty \tag{5.5}$$

$$\lim_{n(q_i)\to\infty} \log_{10} \frac{n(q_i) + 1 + 0.5}{0.5} = \lim_{n(q_i)\to\infty} \log_{10} \frac{n(q_i)}{0.5} = +\infty \tag{5.6}$$

The next part of the algorithm having a significant influence is the query term frequency f (q_i , D), i.e., the times a query occurs in one specific document. A scaling parameter k_1 is introduced to make the query term frequency more or less critical. Again, by using an example the importance of the (scaled) term frequency and influence for the complete algorithm is explained. Suppose the following values (taken on purpose to simplify a lot of calculations):

- $k_1 = 2$
- b = 1
- |D| = 100
- avgdl = 100
- N = 100
- IDF("wing") = 0.94
- f("wing, Document 1) = 5
- f("wing, Document 2) = 20

The two documents are equal in length, and both contain the word "wing", five times in the first and twenty times in the second document. Now in Equation 5.7 and Equation 5.8, the BM25 score is calculated for these documents:

$$BM25(D_1,'' wing'') = 0.94 \cdot \frac{5 \cdot 3}{5 + 2} = 2.01 \tag{5.7}$$

$$BM25(D_2,'' wing'') = 0.94 \cdot \frac{20 \cdot 3}{20 + 2} = 2.56 \tag{5.8}$$

Two remaining parameters, k_1 and b, are to scale specific parts of the BM25 algorithm and make these parts more or less important. The k_1 parameter is to scale the document frequency. The literature recommends a value between 1.2 and 2 when collection specific optimization is not possible. The b parameter is to normalize the document length, necessary because the BM25 algorithm tends to give longer documents, i.e., documents consisting of more words, a higher score. Without collection specific optimization a value of around 0.75 is recommended (a value of 0 corresponds to no document normalization, while a value of 1 means fully scale by the document length).

Python calculates the document length by using the "counter" library, making it easy to calculate the total amount of words in a document. From this, it is easy to calculate the average document length (in words) by dividing the total number of words by the total number of documents.

5.1.7. Feedback

As indicated in the literature study there are three forms of feedback: direct-, indirect- and pseudo-relevance feedback. It is chosen to use the pseudo-relevance feedback for the experiment with as main reason this type of feedback is less dependent on users and external resources.

The procedure of pseudo-relevance feedback is described below:

1. Run original script and calculate BM25 score of documents relevant to the user query.
2. Select N number of top ranked documents.
3. Take M number of top ranked words, calculated with TF-IDF.
4. Add M number of words to original query.
5. Run original script again and calculate BM25 of original queries + added words.

During the experiment, the values of N and M will not change between the problem cases. The number of documents assumed to be relevant is given as five and the number of relevant words is fixed at a value of ten. It is decided to test the effect of changing N and M values during the sensitivity test described in Chapter 5.3. For the user it is possible to toggle feedback on and off by a simple press on the slider, shown in Figure 5.3.

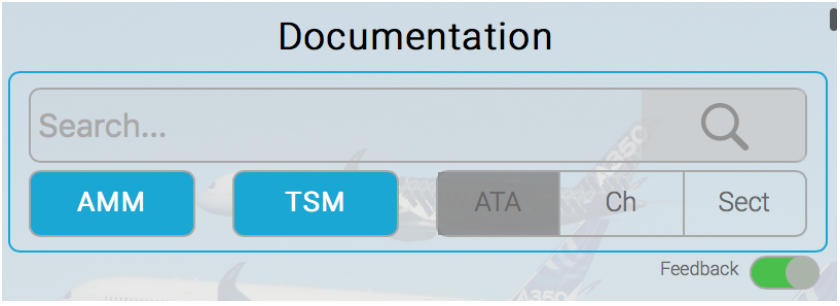


Figure 5.3: New searchbox with slider to turn feedback on/off shown in the lower right corner.

5.1.8. Search Results

The final part is to show the users the documents considered to be the most important to his search. All relevant documents are shown on the left in Macster in decreasing order. Documents with the highest score are shown on top to minimise the necessity of scrolling through many documents. In Figure 5.4 an example of the search result is shown. In the right box, the user is able to scroll through the specific document.

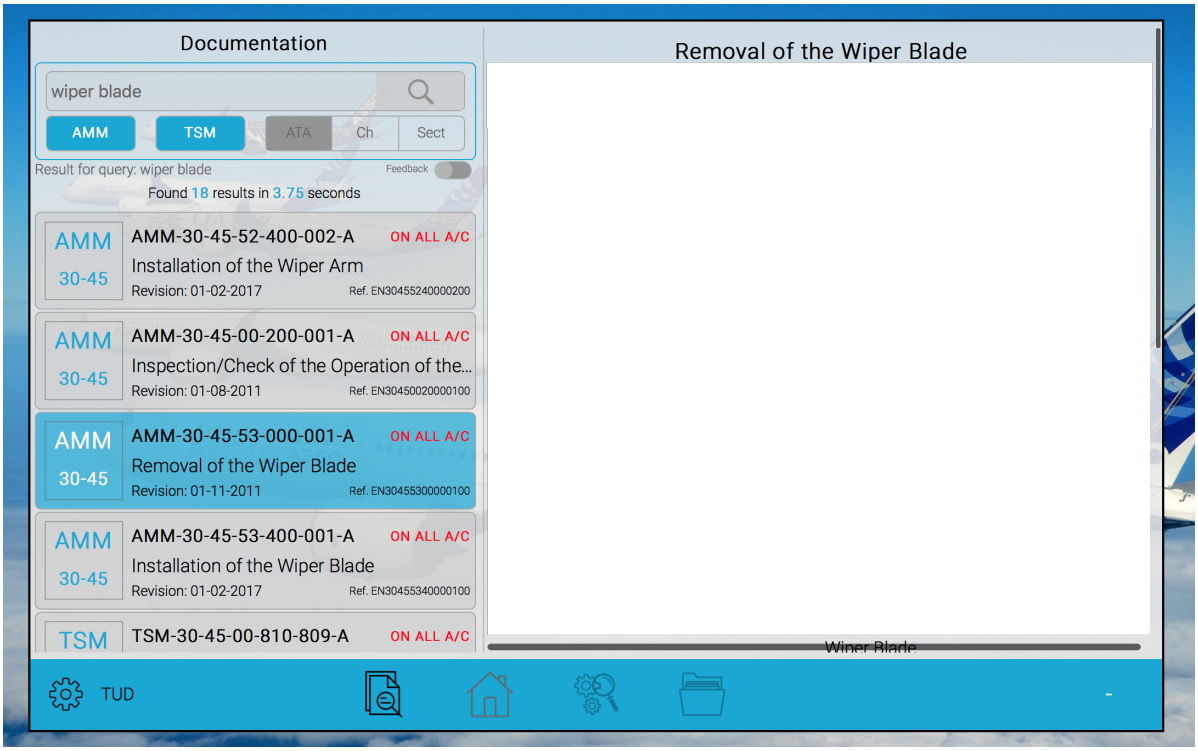


Figure 5.4: Example of search results (content is removed from picture).

5.2. Verification

To verify the system is built correctly, it is necessary to assure the scripts are correctly coded. The main building blocks of the Python code are displayed in Figure 5.1 in the previous chapter. The working of those building blocks will be verified in the subsequent sections. During the verification phase, the results from the Python scripts are compared to manual calculations or visual checks. In some cases, the number of documents used to verify is lower compared to the number of documents used during testing. The reason is to make the manual calculation easier or even doable. Lowering the number of documents has no influence on the trustworthiness of the verification. For Macster, only the document score of the BM25 algorithm and their corresponding ranking score are verified. When the document score and place of the document after ranking are equal to the ones in Python, one can conclude the program works as expected. This because Macster only serves as a mean for the visuals and the Python script is still running in the background (on the server).

Note: because a non-disclosure agreement is signed, it is not possible to show the actual AMM and TSM content.

Documents:

The total number of documents is verified by comparing the number of documents in the local folder on the laptop to the counted number of documents in Python and Macster. First, the 151 maintenance documents are placed in the local folder. Using the print statement, it was found all manuals were also correctly found in Python. The next step is to check if also Macster is able to find all the 151 manuals. To do this, it is necessary the AMM button is turned on and the TSM button is switched off. Figure 5.5 shows Macster is able to show all the AMMs.

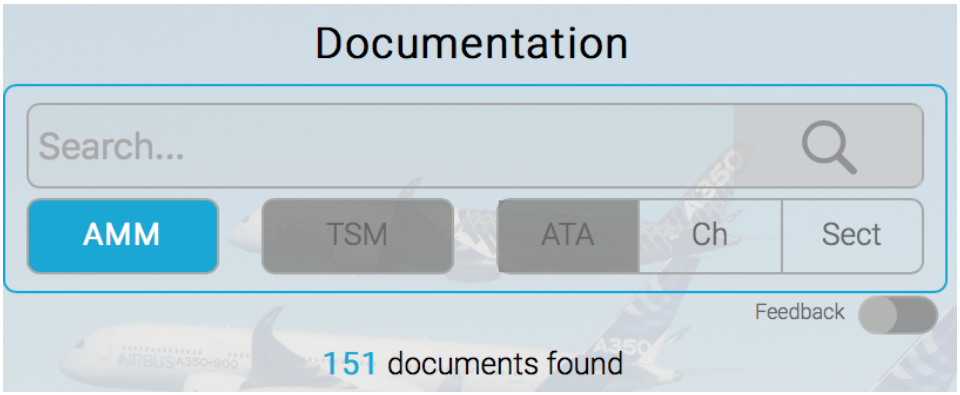


Figure 5.5: All 151 AMM are found by Macster.

The same process is repeated for the TSMs and once more for the AMMs and TSMs together. The results of the TSM search in Macster are shown in Figure 5.6 and the results of the search for all documents in Figure 5.7. For all cases, it turns out the number of documents in Python and Macster is equal to the once stored in the local folder. Therefore, it can be concluded both Python and Macster can find all the documents.

Although Macster makes use of the Python script, it is necessary to do an extra check on the number of documents shown in Macster. The documents used in Macster are stored onto the web server, while the (identical) documents for Python are stored on a local drive to become less dependent of the (private) web server during the system development.

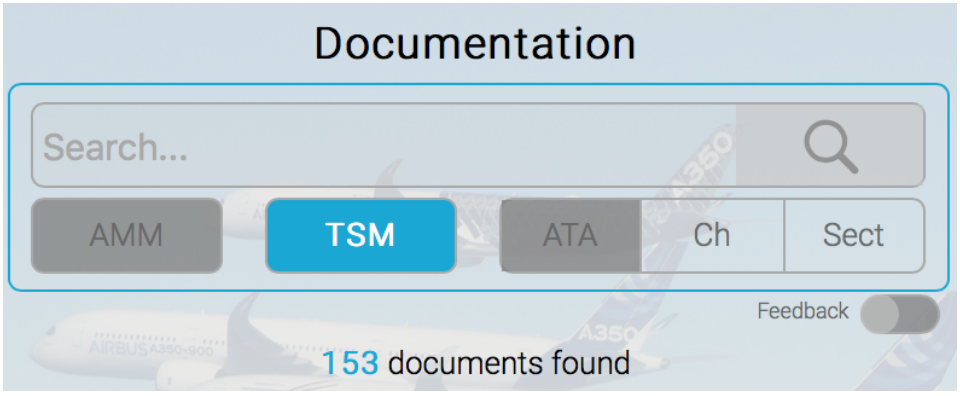


Figure 5.6: All 153 TSM are found by Macster.

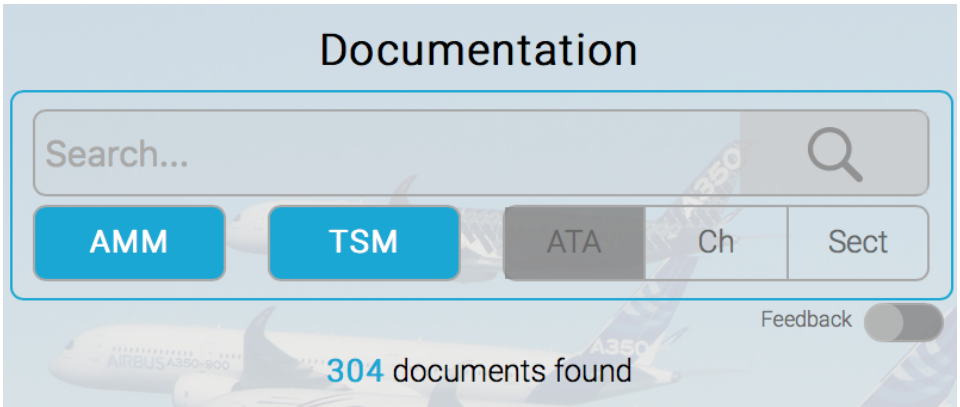


Figure 5.7: All 304 documents (151 AMM + 153 TSM) are found by Macster.

Crawling:

Crawling the documentation is checked by inspection and easily verified because the documents are both readable in Python as in Macster.

An extra check is performed to see the working of crawling. Searching for specific words in the local folder, count them and compare the word count to Python and Macster verifies the crawling function. An example of this word search and word count is shown below. In the local folder, one searches for the word "wing" between the TSMs. This is shown in Figure 5.8 where the search function of the laptop is used, showing the word "wing" is found eight times. The same search is performed in Python, where "wing" is also found eight times in the TSMs, shown in Figure 5.9. Finally, Figure 5.10 shows this search in Macster which concludes this verification.

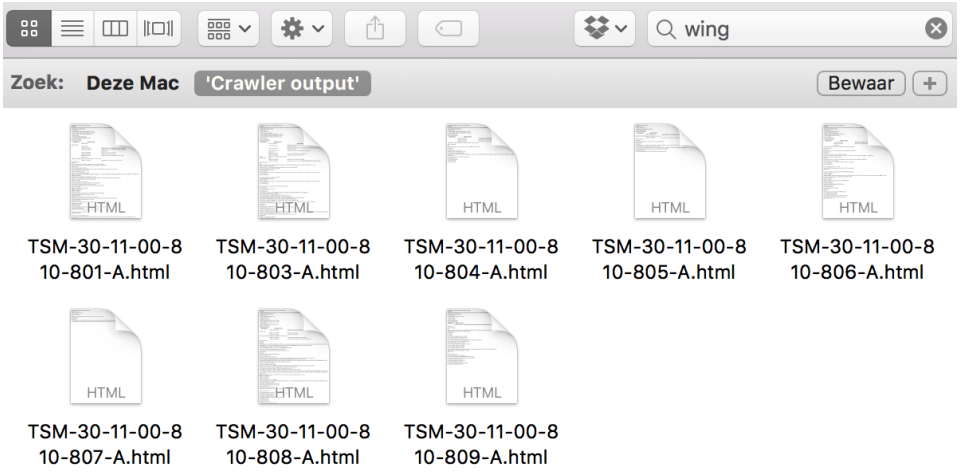


Figure 5.8: The word 'wing' is found in eight documents by the laptop's search function.

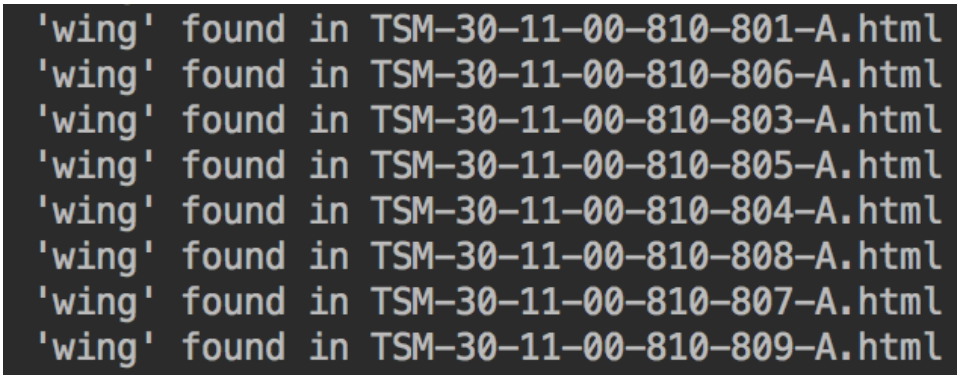


Figure 5.9: Print in Python to show all the TSMs containing the word 'wing'.

Filter / Cleaning:

Verification of the developed filter function is done by visually inspecting the list of unique words. When unexpected characters, tags or punctuation are found, these are added to a list Python needs to replace. Due to time constraints, it is impossible to check for all documents (304 in total) if all tags, punctuation or typos are replaced. It might be possible not everything is filtered out. This will then result in words or characters are seen as unique words in the calculation for the BM25 score. It would cost too much extra time to check every single word. Although not ideal, the effect is minimal because the visual inspection through the inverted index did not show any irregularities. An option is to implement a library and compare the words from the documents with words in the library. Unfortunately, no library is found designed explicitly for technical aircraft documentation. A small test with a library of general words showed most technical words

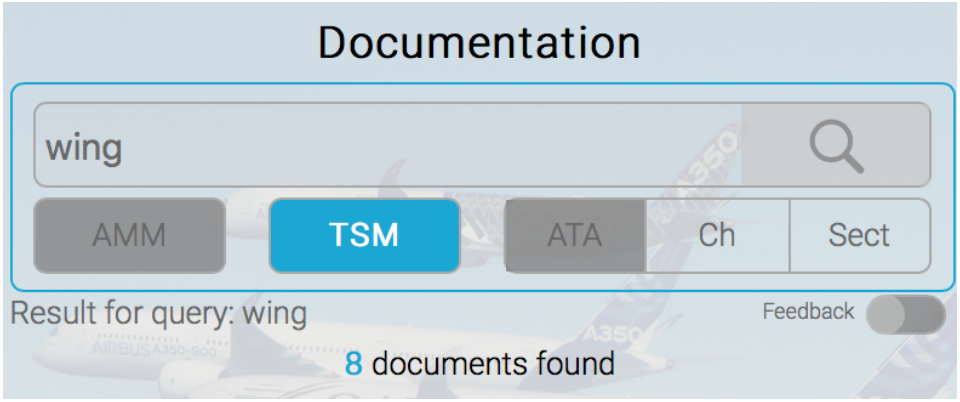


Figure 5.10: 'wing' is found eight times in Macster in the TSM selection.

were not recognized. Another option was to develop a specialised library or to add the technical words from the documents to this library. This option costs time and no guarantee can be given the words are filtered out unfairly. It is decided to take the risk of possibly ending up with some non-unique words left in the inverted index.

Indexing:
An inverted index is created to avoid Python and Macster running through all documents for each search request. The inverted index is a list which stores all words followed by the document name where the word is found. By using an inverted index the ranking algorithm can find specific words in the list without going through all documents for every search. The appropriate structure of the inverted index is defined as: invindex = ['word', 'document name where the word is found', 'number of times the word is in the document']. To check if the inverted index is coded properly three tests are performed. In the first test a document named 'test.html' containing the simple sentence 'this is a test' was added to the folder. After running the script the inverted index is printed. The result of printing the inverted index is [[this, Test.html, 1], [is, Test.html, 1], [a, Test.html, 1], [test, Test.html, 1]] and is considered as correct.

In the second test, the implemented word counter in the inverted index is tested. The word "test" is once more added to the test.html document resulting in the text: "this is a test test". Again the inverted index is printed resulting in [[this, Test.html, 1], [is, Test.html, 1], [a, Test.html, 1], [test, Test.html, 2]]. As can be seen, the value for "test" is increased from 1 to 2 and with this, the function of the word counter is verified.

The final test for the inverted index is to check if it is able to loop over multiple documents. Verified by adding a new document named 'Test2.html' to the folder containing 'this is another test'. The result of printing the inverted index is: [[this, Test.html, 1], [is, Test.html, 1], [a, Test.html, 1], [test, Test.html, 2], [this, Test2.html, 1], [is, Test2.html, 1], [another, Test2.html, 1], [test, Test2.html, 1]] and is considered correct. *To simplify coding in a later stage it is decided to not combine everything together for every unique word, e.g. [this, [Test.html, Test.html2], [1,1]].*

The functionality of the inverted index is now verified with these simple documents. As a next step, the working is tested in Python using all available TSM documents [1] The result of a search for the word "engine" is shown in Figure 5.11 (after some visual manipulation to make it readable, which did not affect the verification process). The search function of the laptop is used as a verification method to show each document also contains the word "engine" as many times as the Python script indicates. Again but now for all TSMs, the working of the inverted index is verified.

The Python script has the possibility to find words containing part of a query, although this functionality will not be active during the experiment. Some simple verification is done by typing part of a word as a search query. A search of the query "is" is performed on the previously described documents test.html and

[1]The reason to verify the working of the model using the TSMs and not the AMMs is because at the time starting the verification phase, the AMMs were not locally available.

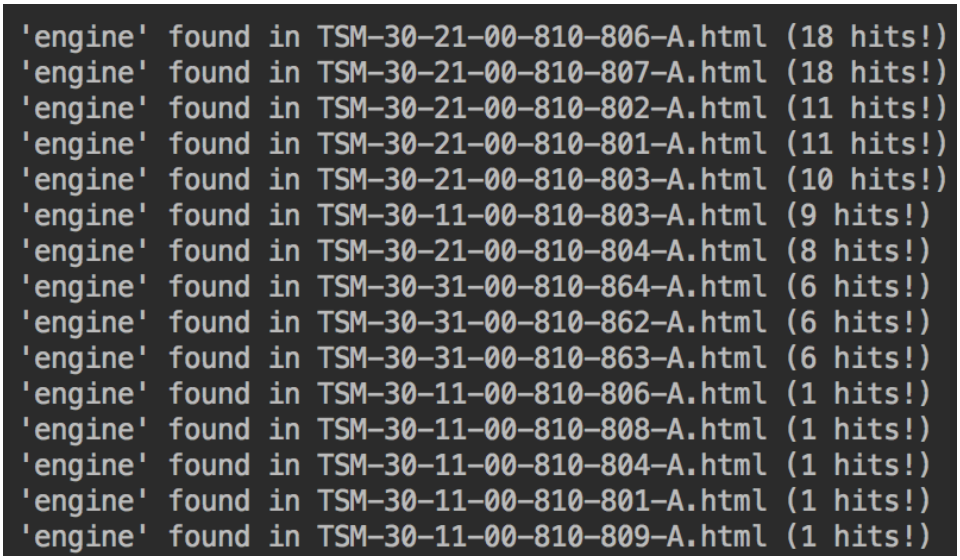


Figure 5.11: Elements of the inverted index in Python show the search result of "engine", the document name and the number of times it was found.

test2.html resulting in an output of: "this" and "is". The output is correct since the word "is" is part of "th**is**". Because only the Python function "in" is used, which is part of the standard Python package, it is considered as unnecessary to further verify if this function works correctly.

Ranking:

The verification of ranking is done by inspection of the individual BM25 elements. The algorithm used is given by previously shown Equation 5.1 and Equation 5.2. In the end, the total score of each document is verified and the scores resulting from Python are compared against the Macster scores. The score of a document for a query is dependent on the number of words and the total number of documents. It is therefore a small number of documents is used for the manual verification.

Inverse document frequency (IDF): The inverse document frequency consists of the number of documents in the collection (N) and the number of documents containing the specific search query n (q_i). The total number of documents in the collection is already verified. Documents containing specific queries are indirectly also verified in the indexing section. In here, for a specific query, the document where the query is found is stated (including the number of times it was found). Taking the sum of each document where the query was found results in the number of documents with this specific query. Multiple queries are searched for in the TSM documents, to verify this all goes correct. The document score in Python is compared to a manual calculation. Table 5.1 shows an overview of the theoretical scores and scores resulting from Python. The scores are equal for all queries and with this the working of the IDF calculation is verified.

Query	**n (q_i)**	**Theoretical IDF score**	**IDF score Python**
Engine	15	0.95111807523	0.95111807523
Wing	8	1.23344406761	1.23344406761
Valve	15	0.95111807523	0.95111807523
Filter	2	1.78247262417	1.78247262417
Aqua	0	2.48713837548	2.48713837548

Table 5.1: Overview of queries and corresponding theoretical IDF scores and IDF scores calculated by Python (solely TSM are used, N=153).

Query term frequency (f (q_i , D)):
The query term frequency (f (q_i , D)) is already verified in the index section.

Total number of words per document (|D|):
The total number of words per document is verified with the test.html and test2.html file, from which the word count is considered as correct. The words from one TSM are manually summed up and compared to the total number of words per document in Python. It is found the results from the manual calculation are identical to the ones in Python.

Average number of words for all documents (avgdl):
The average number of words for all documents is simply the summation of all words for each document divided by the total number of documents. The result from the two small test files is that the average word count was 4.5 and considered correct. During testing the amount of documents and the documents itself does not change and therefore the average number of words for all documents could be programmed as a fixed value. To keep the options open for expanding the model the value is not fixed and calculated every time one runs the script.

BM25 Score: First the scoring of the documents is verified. This is done by comparing a manual calculation by the calculations performed in Python. A summation is included in the BM25 algorithm and therefore this part is also verified, by comparing the sum of multiple separate queries in one document with the search of all these queries in one search.

Feedback: To verify the feedback algorithm some extra steps are required. First, the IDF function is verified in the same way as described in the example in section 5.1.6. The working of the IDF function is further verified by comparing the value of a search when no document is found containing a specific query. In case all the 304 documents are available in Python this would result in an outcome of 2.7846 ($Log(\frac{304+0.5}{0.5}) = 2.7846$). After performing a search on "clown" (a word which certainly does not exist in one of the manuals) the IDF value was indeed 2.7846. Next, it is verified with a simple print function Python indeed takes N number of files and M number of words. All the previous is working correctly and from here the same BM25 algorithm runs through but now with the addition of the extra M words of the top N files.

5.3. Sensitivity Analysis

In the IR domain, the optimisation of parameters is very expensive because it requires a human evaluation of many query results, which are also specific to the collection. Often one sees the optimisation procedure also as computationally costly because it requires more computing power than the search engine itself. The purpose of the thesis is not to deliver the best-ranking algorithm, however, to be aware of the effect of each parameter is always preferable. To see this effect, in this section a sensitivity analysis is performed on multiple parts of the algorithm of both BM25 and feedback. In the end, the main conclusions are given about the findings.

To fairly compare the results from this sensitivity analysis, specifications of the equipment are given first. This analysis is performed on a Mid 2014 MacBook Pro with the following specifications:

- 2.5GHz quad-core Intel Core i7 processor (Turbo Boost up to 3.7GHz) with 6MB shared L3 cache
- macOS High Sierra 10.13.6
- 16GB Memory
- 512GB Solid State Drive
- Pycharm 2016.3.3

The server is a hosted server from cloudvps and has the following specifications:

- 1 GB Core
- 2.5 GHz Xeon CPU

The sensitivity analysis is performed with only the AMM documents in the collection, unless mentioned otherwise.

Time to create index

One part of the research question is to find the efficiency impact on the search results due to the implementation of the BM25 algorithm. Efficiency relates to the time it takes to find the appropriate results. The system needs time to deliver results and therefore the first sensitivity analysis is performed on the time to create the inverted index in Python. In Table 5.2 an overview is given of the time to create an index with a certain amount of documents. More importantly, is the time to create an index consisting of a number of words. Dividing the time to create the index by the total words, results in a time to index per word. This can be useful when more documents (and therefore more words) are added to the collection. It is found the time to create the index is around the $6.5 \cdot 10^{-6}$ seconds/word. The exact time to index is dependent on the script but also depends on the computer or server used. For every configuration, it is possible to calculate the value of time to index per word.

Number of documents	Total words	Index size (unique words)	Time to create index (s)
1	2285	292	0.084
5	7769	362	0.124
10	17777	650	0.174
50	84362	1632	0.584
100	147194	2247	1.063
151 (all AMM documents)	222970	2805	1.476
304 (all documents)	350672	3390	2.252

Table 5.2: Time to index documents.

Variation in search time for same queries BM25 algorithm

The next step is to investigate whether the system shows significant variations in search time for the same queries. In Python and Macster several queries are looked up for multiple times and the time it takes to return the results is written down. For this analysis, all documents are used (AMM + TSM). An overview of the Python results is shown in Table 5.3 and the results from Macster in Table 5.4.

Query	Time (s) run 1	Time(s) run 2	Time (s) run 3
test	2.322	2.338	2.330
wing	2.278	2.309	2.341
valve	2.306	2.296	2.300
operational	2.329	2.335	2.297
air	2.355	2.327	2.310
intake	2.284	2.300	2.267

Table 5.3: Time to return result for same query in Python without feedback.

The first to notice in Table 5.3 is the variation in search time is in the hundreds of seconds. This is true for the time to find the same queries, but also to find any other query. Therefore one can say the Python results are very constant. Secondly, it can be mentioned the time to find the query is the inverted index is extremely low (in the best case around $1 \cdot 10^{-2}$ second). Both cannot be concluded from the results of Macster in Table 5.4. In the worst case, the variation is more than a full second. Also, the time to return the results is in most cases twice as high as the Python results. One reason for this is possibly the use of the one core server, which takes a longer time to perform the calculations of the document score. Another reason for the extra time is the conversion of the queries from Macster into the Python script and back again for the ranked documents to show. Thirdly, the connection speed (ping) and the internet quality (packet loss) influence the retrieval time for Macster.

Query	Time (s) run 1	Time(s) run 2	Time (s) run 3
test	5.83	3.65	5.50
wing	5.62	5.48	5.16
valve	4.93	4.32	3.59
operational	3.93	5.40	5.00
air	5.30	4.57	5.87
intake	5.51	5.67	5.72

Table 5.4: Time to return result for same query in Macster without feedback.

Variation in search time for same queries BM25 algorithm and feedback

The same analysis as described in the previous section is performed, but now with feedback enabled. The feedback procedure requires extra calculations and therefore it takes longer for the model to retrieve the most relevant documents. In Table 5.5 the Python results are shown and in Table 5.6 the results from Macster. Again the time to run the Python script and return the results are stable per query and only vary by tenth of seconds. Unfortunately, for different queries there can be almost a factor two difference which is due to the feedback procedure. The difference between the queries is due to the fact different documents are retrieved for the feedback procedure. For some queries, it is required to search through many documents and many IDF calculations must be performed to determine the most relevant words. Other queries might be specific and fewer documents and/or shorter documents have to be searched. The problem is that it is impossible to know beforehand which documents are searched through. What is known, is the speed to perform the feedback calculations has a major impact on the time to show the results. As expected, the results from Macster are even worse. Not only the variation in retrieval time for different queries can go up to almost a factor four, also when the same query is put into the system there can be a difference of more than three seconds. The variation in different retrieval times for different queries has similar causes as for the differences found when applying the BM25 algorithm without feedback. The variation in retrieval time using the same query is strange and expected to be due to internet connection or server performance.

As an extra check, the time before going to the feedback procedure was printed in Python and for all queries this was, as expected, to be around 2.3 seconds.

Query	Time (s) run 1	Time(s) run 2	Time (s) run 3
test	4.875	4.864	4.757
wing	5.199	5.195	4.915
valve	7.636	7.528	7.852
operational	4.173	4.074	4.098
air	5.839	5.841	5.750
intake	4.588	4.767	4.747

Table 5.5: Time to return result for same query in Python with feedback (N=5, M=10).

Query	Time (s) run 1	Time(s) run 2	Time (s) run 3
test	14.96	14.19	15.77
wing	9.85	12.99	10.25
valve	11.67	10.94	13.55
operational	18.79	15.72	19.19
air	6.63	7.88	7.07
intake	8.06	7.76	5.23

Table 5.6: Time to return result for same query in Macster with feedback (N=5, M=10).

Variation in search time when number of queries increases

In Table 5.7 and Table 5.8 the results are shown when the number of words of the query is increased in Python and Macster. The table shows both the results of the algorithm without feedback and including feedback. The addition of extra words does not affect the time to deliver results without feedback. Remarkable to see is

the addition of more queries in Python has a positive influence on the time to deliver results when feedback is enabled. The results from Macster show again significant fluctuations.

Queries	Time (s), no feedback	Time(s), with feedback
test	2.279	4.969
test wing	2.352	4.252
test wing valve	2.294	4.158
test wing valve operational	2.345	3.978
test wing valve operational air	2.334	3.967
test wing valve operational air intake	2.361	3.943

Table 5.7: Time to return search results for increase in number of queries for Python

Queries	Time (s), no feedback	Time(s), with feedback
test	5.60	16.50
test wing	5.46	9.27
test wing valve	5.41	11.83
test wing valve operational	4.22	9.02
test wing valve operational air	3.66	10.67
test wing valve operational air intake	4.42	9.87

Table 5.8: Time to return search results for increase in number of queries for Macster

Comparing the results from Table 5.7 and Table 5.8 shows some interesting facts. First, the time to return the results to the user is for a regular search around a factor two higher for Macster. However, by enabling feedback the time to return the results increases by to almost a factor three. Secondly, for the results in Python, it costs only around two additional seconds to perform the calculation with feedback, while to return the same results in Macster, the difference is between five and eleven seconds. The fluctuation in the difference of returning those results is remarkable and further investigation in the future is necessary to provide a clear explanation. For now, it is expected it is due to server performance.

Baseline for different queries

First the rank of AMM 30-21-00-710-002-A is compared when searching with several different queries. In this test only the AMMs are in the collection and feedback is not enabled. The values for the tuning parameters are chosen to be values given in literature (without collection specific tuning), k_1 = 1.5 and b = 0.5. The used queries are guesses from ones mechanics might use. [2]

Query	Rank correct document	Python search time (s)
Operational test	9	1.505
Engine test	76	1.521
Perform engine test	76	1.523
Operational engine test	5	1.521
Air intake ice protection	2	1.515

Table 5.9: Baseline for different queries with k_1=1.5 and b=0.5, no feedback and only AMM documents.

Table 5.9 shows the use of different queries results in an enormous difference in rank. This difference in rank, 74 positions, is due to the fact some words are used very often and therefore they have a low BM25 score, e.g. ”test” is found in 144 documents (from the total of 151).

[2] In Table 5.9 there are no results shown for Macster. This because at the time of the analysis of Table 5.9 there was no access to Macster. Therefore it was impossible to change the available documents Macster can use during the calculation. This is important to mention because, in standard conditions, Macster uses both AMM documents and TSM documents. The rank of the document, however, does not change compared to Python.

For now it can be said for this problem, with only AMMs and feedback is not used, the correct document is found at the highest rank of two and the lowest of 76.

The next test is to find out what the addition of the TSMs to the collection will do to the results using the same search queries and tuning parameters. The results can be found in Table 5.10.

Query	**Rank correct document**	**Python search time (s)**	**Macster search time (s)**
Operational test	105	2.343	5.53
Engine test	35	2.275	5.52
Perform engine test	35	2.340	5.63
Operational engine test	35	2.326	5.38
Air intake ice protection	8	2.346	3.91

Table 5.10: Baseline for different queries with k_1=1.5 and b=0.5, no feedback and both AMM and TSM documents.

Immediately visible is the highest rank is lowered from two to eight and the relatively good scoring query ”operational test” is dropped significantly. The reason for this drop is clear, the word ”operational” is found 32 times in the AMMs, with the addition of the TSMs it is found in a total of 157 documents. This test shows the major influence of the number of documents in the collection and their content. The question is if pseudo-relevance feedback can improve the search results for all these queries or a combination of parameter tuning and feedback is required.

In Table 5.11 the results are shown from the test where the feedback algorithm is enabled but remaining parameters are unchanged and kept constant.

Query	**Rank**	**Python search time (s)**	**Macster search time (s)**
Operational test	110	3.335	16.30
Engine test	63	2.365	11.64
Perform engine test	63	2.510	12.26
Operational engine test	63	2.458	12.25
Air intake ice protection	12	4.454	8.54

Table 5.11: Baseline for different queries with k_1=1.5 and b=0.5, including feedback (N=5, M=10) and both AMM and TSM documents.

From Table 5.11 it can be concluded feedback in this form does not add any value to the results because for all queries the results are worse compared to the results without feedback. Important to mention is that the rank is without making a selection of the type of document.

To see the effect of changing k_1 an b values some additional test is performed. In the first test the b value is kept constant (b = 0.5) and the k_1 value is increased in several steps. From the previous results it can be found that with and without feedback the queries ”operational test” always has the lowest score (worst rank) and the ”air intake ice protection” has the highest rank. This test is performed on all queries and the results are shown in Figure 5.12, for the highest scoring (”air intake ice protection”) and lowest scoring (”perform engine test”) queries the results are shown in Table 5.12.

Query	k_1	Rank correct document	BM25 score
Air intake ice protection	0.01	2	1.734
Air intake ice protection	0.1	2	1.835
Air intake ice protection	0.5	2	2.218
Air intake ice protection	1	2	2.586
Air intake ice protection	1.5	2	2.867
Air intake ice protection	2	2	3.087
Air intake ice protection	10	2	4.056
Perform engine test	0.01	34	-1.087
Perform engine test	0.1	34	-1.136
Perform engine test	0.5	46	-1.308
Perform engine test	1	63	-1.450
Perform engine test	1.5	76	-1.541
Perform engine test	2	80	-1.600
Perform engine test	10	56	-1.553

Table 5.12: Effect of changing the k_1 parameter, where b is kept constant (b = 0.5).

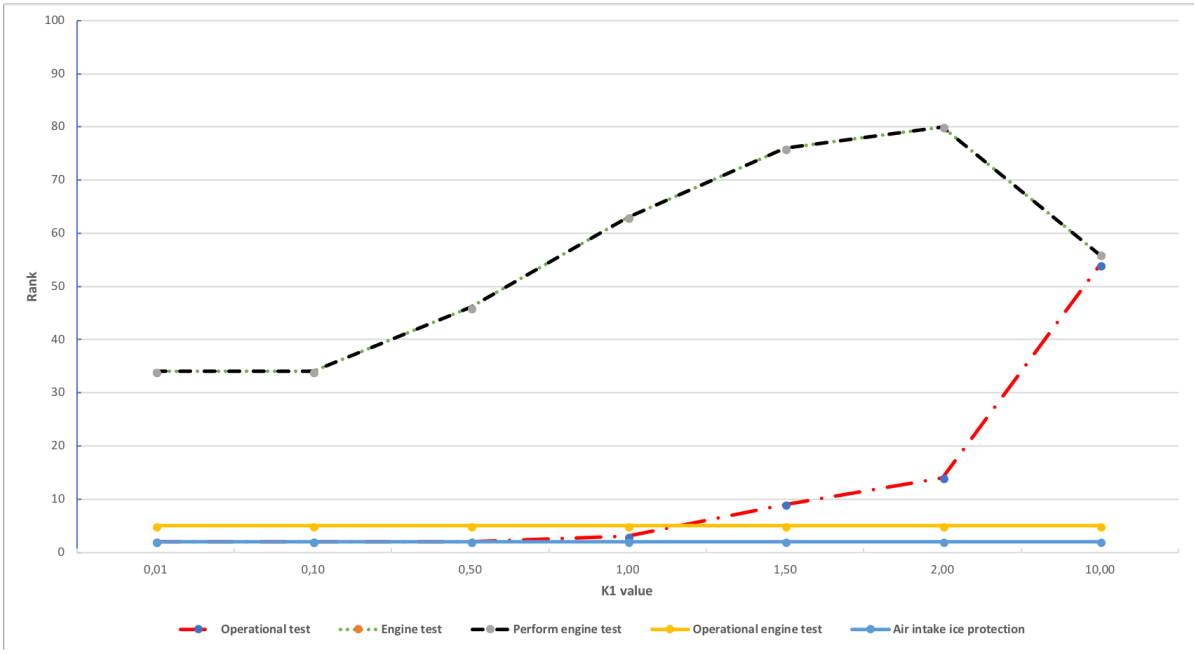


Figure 5.12: Rank of search queries for different k_1 values, where b is kept constant (b=0.5)

From both Figure 5.12 and Table 5.12 one can see the best results are achieved when the k_1 value is low, meaning almost no term frequency scaling. To see if this also holds for the b value a new test is performed, where the b values are changed and the k_1 value is kept constant at 1.5.

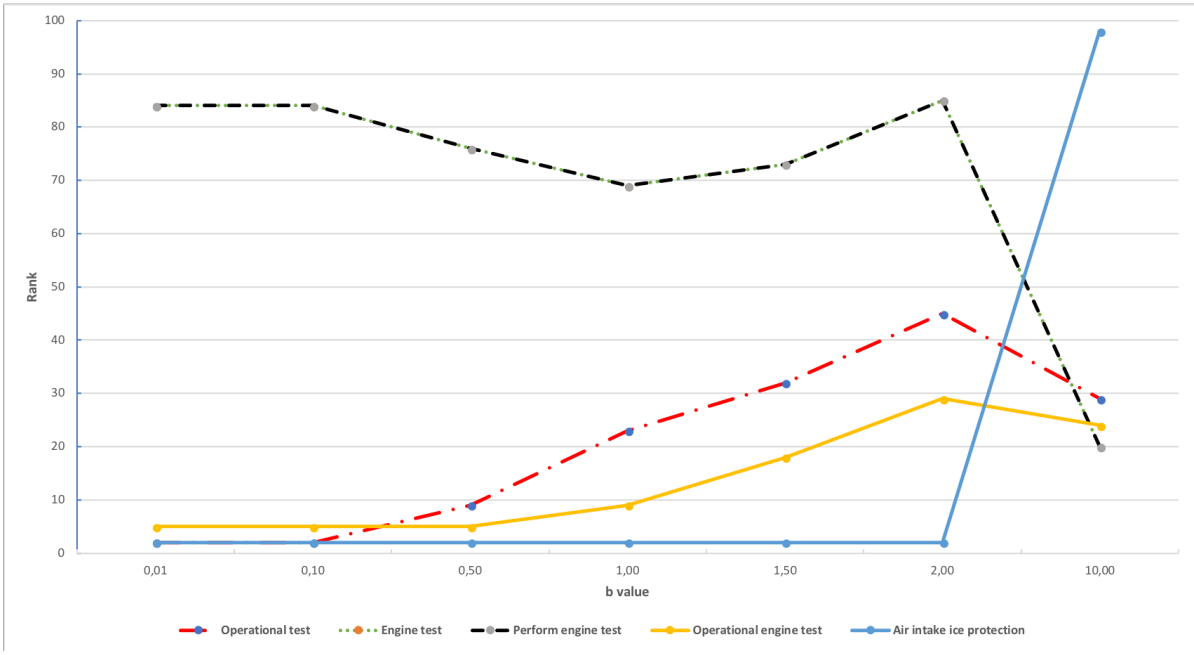


Figure 5.13: Rank of search queries for different b values, where k_1 is kept constant (k_1=1.5)

The results from this test are remarkable, because depending on the query the search results become better or worse. This is the case for a b value starting at 0.1 and continues until a value of 1.

In the final test each query is investigated separately, where both the b value as the k_1 value is changed. For the query "air intake ice protection" it was found the rank is always 2, independent of the combination of b and k_1, therefore the results are not visually shown. The results for the queries "perform engine test" and "engine test" are equal because after evaluation it was found the word "perform" does not occur in any of the documents. It does therefore not lead to an increase in BM25 score, because a score of 0 (from "perform") is added.

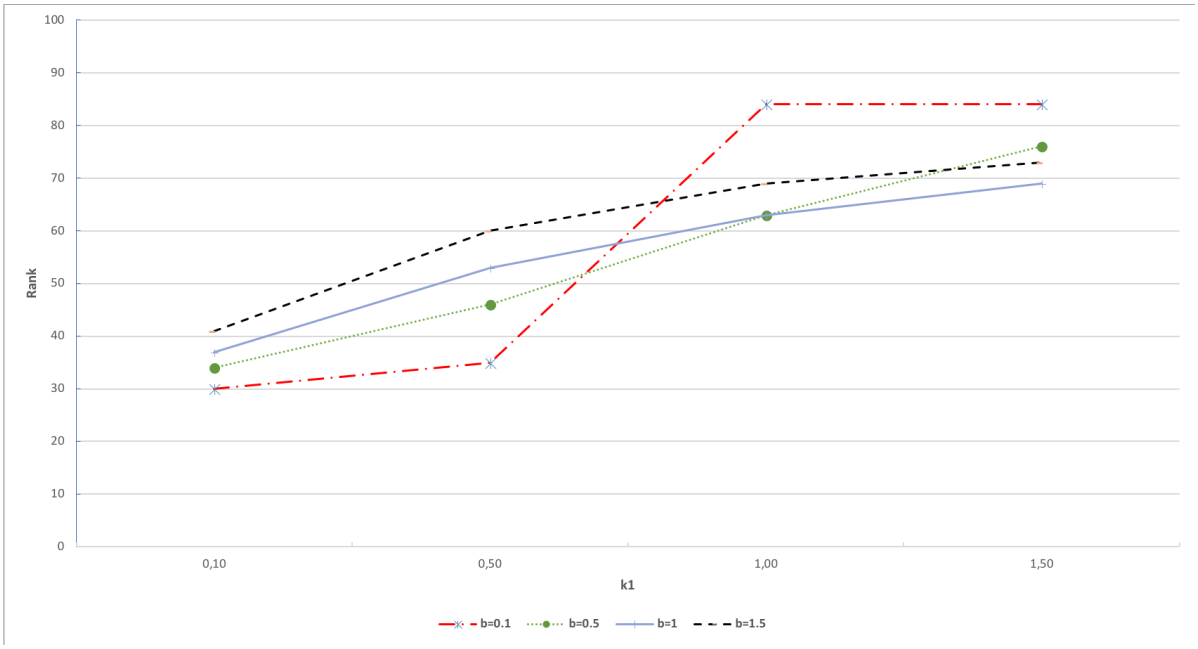


Figure 5.14: Rank of correct document for search queries "(perform) engine test" for different values of b and k_1.

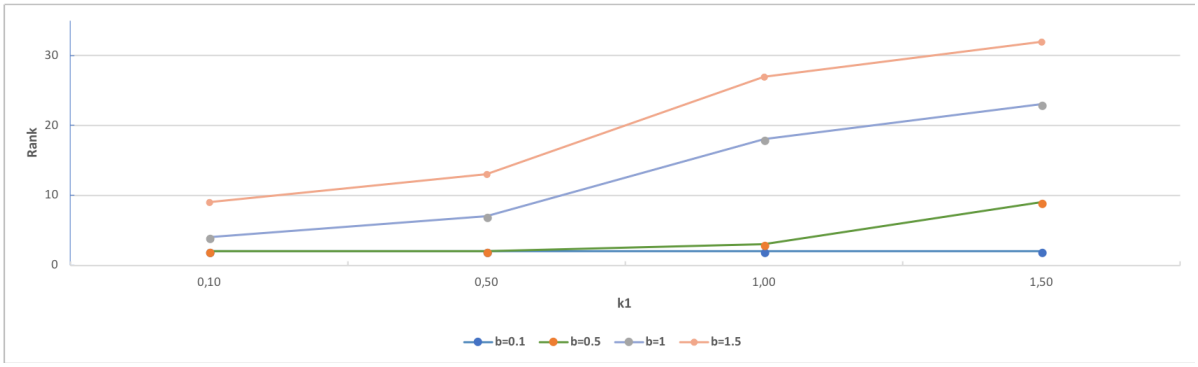


Figure 5.15: Rank of correct document for search queries ”operational test” for different values of b and k_1.

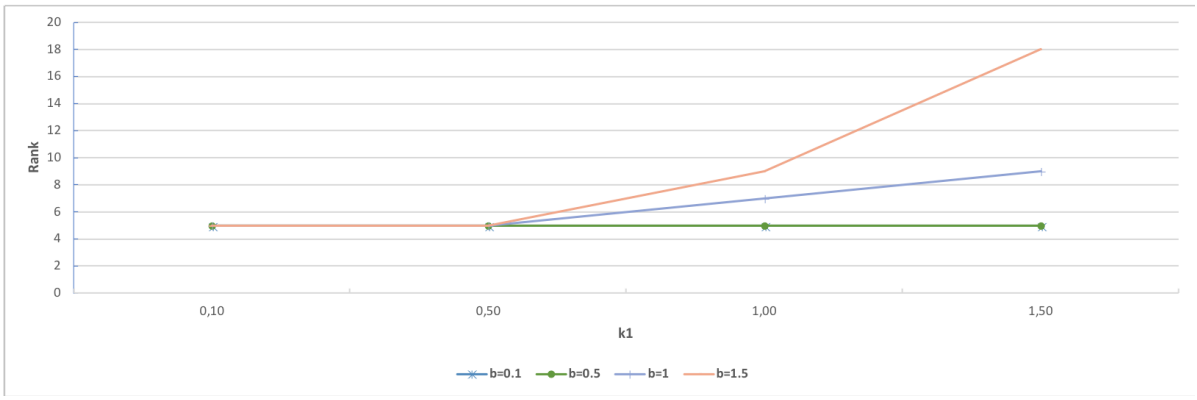


Figure 5.16: Rank of correct document for search queries ”operational engine test” for different values of b and k_1.

Feedback

To investigate the effect of feedback multiple tests are performed. For these tests all AMMs and TSMs are used as part of the collection. The values of b=0.5 and k_1=1.5 are used and kept constant. The same queries are used as the tests without feedback, but now the number of relevant words (M) and relevant files (N) is changed. The results are shown in Figure 5.17 till Figure 5.20.

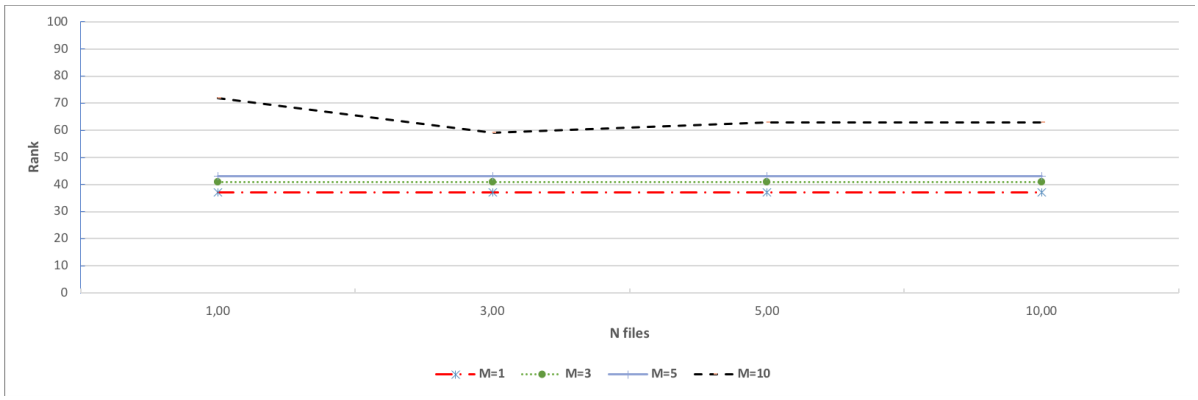


Figure 5.17: Rank of correct document for search query ”perform engine test” where number of relevant words (M) and relevant files (N) is changed.

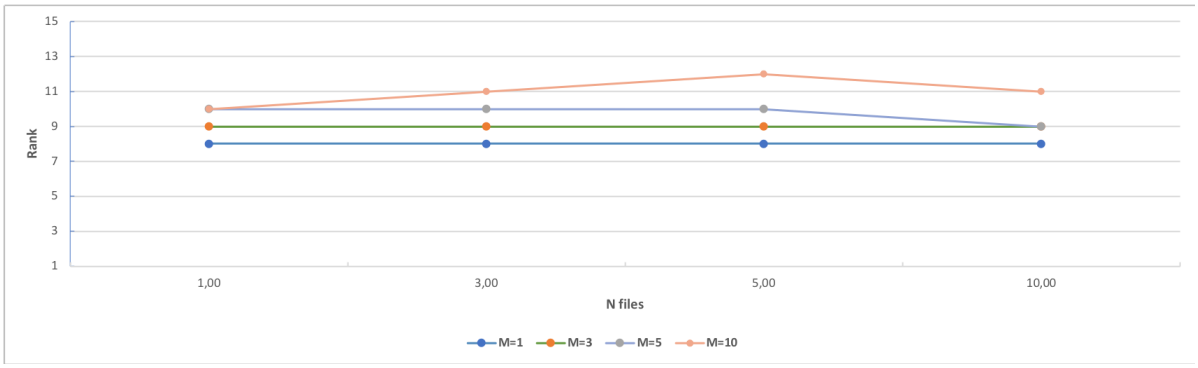


Figure 5.18: Rank of correct document for search query "Air intake ice protection" where number of relevant words (M) and relevant files (N) is changed.

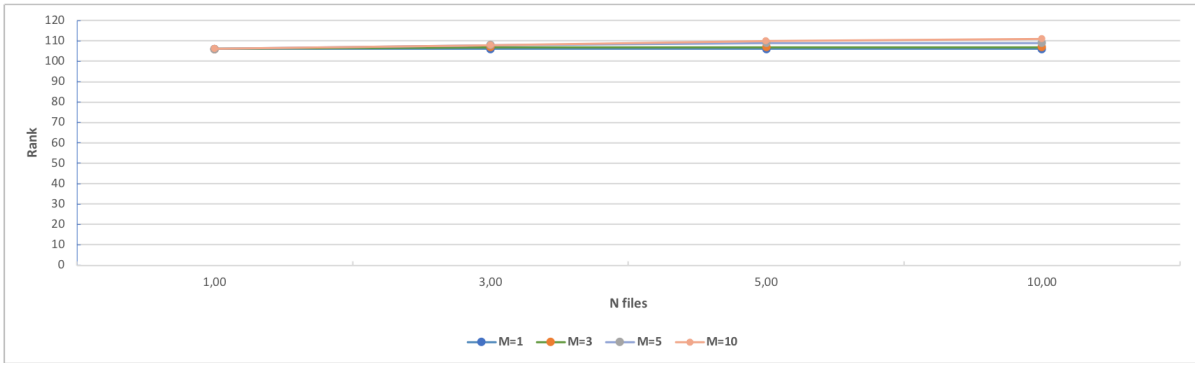


Figure 5.19: Rank of correct document for search query "Operational test" where number of relevant words (M) and relevant files (N) is changed.

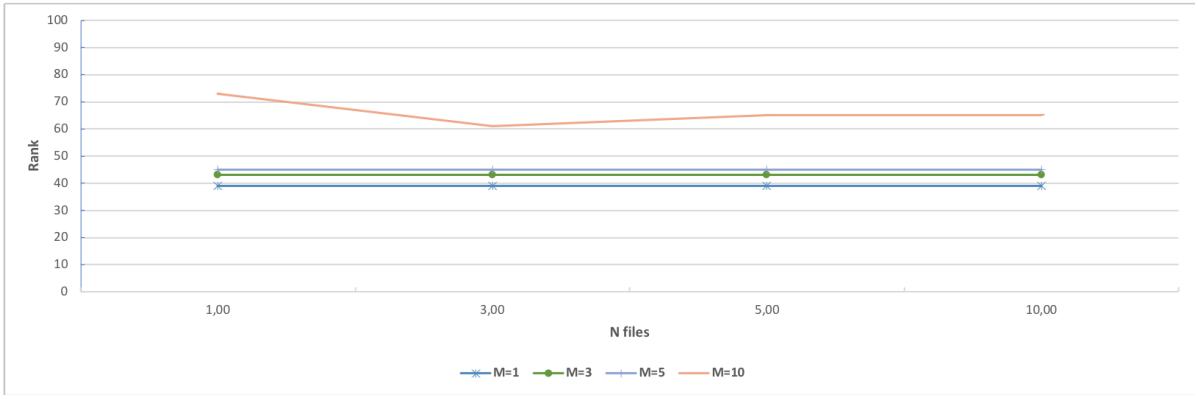


Figure 5.20: Rank of correct document for search query "Engine test" where number of relevant words (M) and relevant files (N) is changed.

The Figures 5.17, till Figure 5.20 make one thing very clear: the addition of extra relevant files (N) does not lead to better results for the queries used. For every test the best results are achieved when only one file is assumed as relevant. The feedback procedure only includes results for the BM25 values of b = 0.5 and k_1=1.5. To make a fair comparison, the results between the two procedures are evaluated only for the same b and k_1 values. In Table 5.13 an overview of each search query and its highest rank is given.The results are mixed and the additional use of pseudo-relevance feedback is not proven at this moment. As indicated earlier, the best results for the procedure without feedback are achieved when the b and k_1 value are as low as possible. Due to time constraints it is not possible to do an extra investigation on the impact of the b and k_1 when feedback is enabled. Therefore this is given as a recommendation for further research explained in Chapter 7.

Query	Highest rank (no feedback)	Highest rank (feedback enabled)
Operational test	9	106
Engine test	76	37
Operational engine test	5	39
Air intake ice protection	2	8

Table 5.13: Highest rank for query with and without feedback, with b = 0.5 and k_1 = 1.5

Verification of user items

The purpose of this research is to show the impact of the BM25 algorithm on relevant document retrieval and corresponding efficiency with and without pseudo-relevance feedback. The verification of the separate model items is performed. However, the user must be able to work with the model. It is unimportant to the user to understand its working, as long as it easily retrieves documentation. The user makes use of Macster and therefore the following items are verified.

- Possibility to search on multiple queries.
- Documents are shown in order (highest ranked documents first).
- The filter option is present and functional.
- Option to enable feedback is present.

The working of the search button must be verified first. It is already shown previously for a one-word search is properly working, but users must be able to search on multiple queries at once and therefore this extra test is performed. It is important the document with the highest score is shown first to the user. Already verified is the calculation of the document score is performed correctly, however it is not shown these documents are displayed in the correct order (highest score first). This verification is performed for multiple queries, but the results from only one query search are shown. In Table 5.14 the top five Python results of using the query "do a visual inspection" are shown including score and document name. The search uses the following parameters: $k_1 = 1.5, b = 0.5$. In Figure 5.21 the results from Macster are shown. The working of this verifies both the working of the use of multiple queries as the visual ordering in Macster.

Document name	BM25 score
AMM-30-71-51-200-001-A	1.7615460299
AMM-30-21-49-200-001-B	1.41931081559
AMM-30-21-00-710-001-A	1.330495845
AMM-30-21-51-210-040-A	1.19443975057
AMM-30-81-21-200-001-A	0.215909384689

Table 5.14: Top five Python results from the search "do a visual inspection".

During the experiment the user does not have to specify the document type because only the AMMs are taken into account. The option to use the TSMs is turned off on the server. The option to use the TSMs is already implemented and it is possible to do a search and filter on document type. The user must be able to turn the feedback option on and off during the experiment. To verify the feedback option, the same query is used, but now with only the AMMs and feedback enabled. For feedback the following values are chosen: N = 5 and M = 10. The results are shown in Figure 5.22 and it is found the feedback mechanism indeed changes the order of documents.

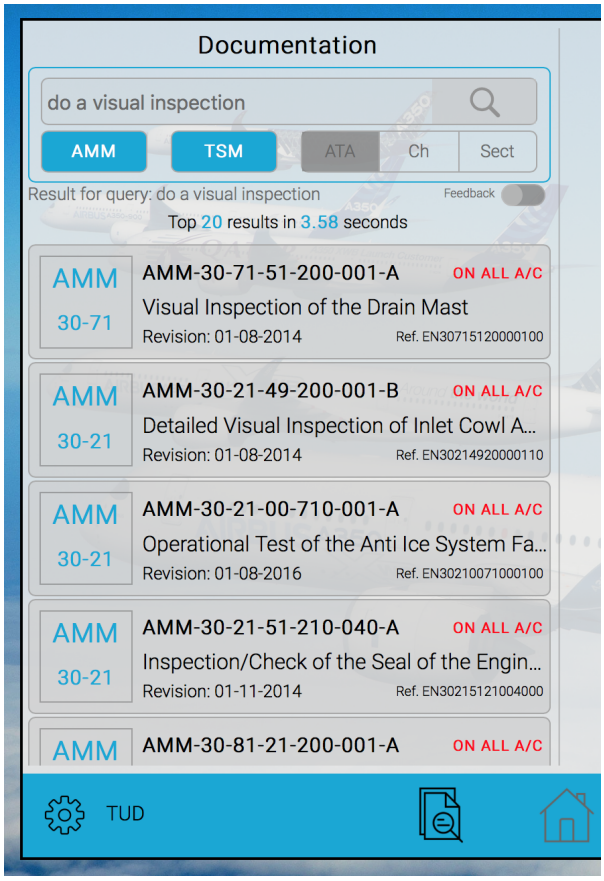


Figure 5.21: Macster showing the ordered results from the search "do a visual inspection".

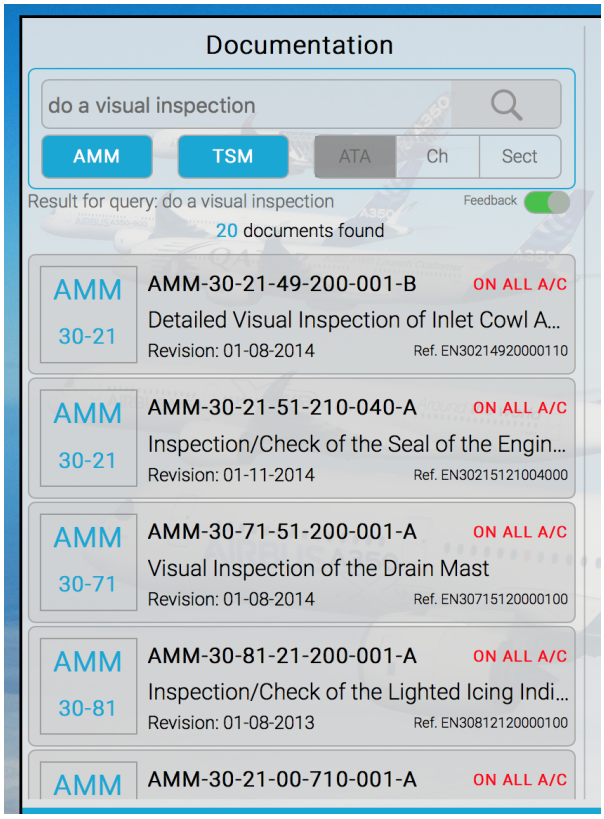


Figure 5.22: Macster showing the ordered AMM results from the search "do a visual inspection" with feedback enabled.

6

Test Results and Discussion

This chapter discusses the results of all tests conducted by the AMTs. The first three sections show the results for the individual problems using all three methods, e.g., the PDF search, BM25 algorithm and the algorithm in combination with pseudo-relevance feedback. The results from these tests are bundled together and discussed in the final section. Packing the results together makes it possible to see how each method behaves in general. Although six tests do not entirely represent a real-life system behaviour, it will give a good first indication.

Although the results from the sensitivity analysis show the use of b and k_1 values to be as low as possible, values of b=0.5 and k=1.5 are used for all tests. Due to time constraints, it was impossible to shift the tests to a later stage and at that point in time the sensitivity analysis was incomplete. The b and k_1 values are chosen to be close to the ones given in literature and the results from Figure 5.13, which was already complete, made it a valid choice.

6.1. Problem 1

Table 6.1 shows an overview of the results from the first problem with a division in the A and B case. Figure 6.1 shows the number of AMTs able to find the correct document including the average search time to find these documents for the first problem in total. The search time of AMTs who were not able to find the correct document is not taken into account in the average search time.

Figure 6.1 shows that 27 out of the 34 AMTs were able to find the correct AMM, with an average search time of 228 seconds for a search using the PDF manuals. This average time is calculated by dividing the total time all AMTs needed to solve problem 1A and problem 1B, by the number of AMTs able to find the correct document, which was 27 in the case of using PDF manuals. The mentioned 34 AMTs are the result of two tests (A + B) with the 17 participants. Not shown in this table, but found as the peak value in Figure 6.3 is one AMT needed 529 seconds to find the correct document. There were also three AMTs who after 575 seconds, 612 seconds and 833 seconds came to the conclusion they were not able to find the correct document (as mentioned these results are not accounted for in Figure 6.1). Table 6.1 does not include iterations for the PDF search because the AMTs were not forced to make use of a query search. Figure 6.3 shows the distribution of search time for a PDF is equally spread, there are no outliers and is (slightly) right skewed. The figure shows the variation in average search time is large, which supports the fact that AMTs are sometimes unable to find the correct documentation on time.

Looking to the results of the BM25 algorithm search it is found that, with and without feedback, 32 AMTs were able to find the correct document. This shows independently of feedback, 15% more AMTs were able to find the correct document compared to the current way of working (PDF search). A closer look shows the use of the BM25 algorithm is able to reduce the search time from 228 seconds to 62 seconds, an efficiency increase of 27% for this specific problem. When also feedback is applied, the average search time is decreased to 41 seconds. Figure 6.2 shows for the BM25 + pseudo-relevance feedback method, two AMTs more were able to find the correct document after the first iteration compared to the normal BM25 situation. An iteration

is when a query did not deliver the expected result and a new query is used to find the same document. Figure 6.3 shows for both methods the spread in average search time is considerably less compared to the PDF search, meaning not only on average the AMTs spend less time in finding relevant documentation. For the BM25 results, one outlier is found from an AMT who needed one iteration. This is remarkable because one should expect it came from an AMT with three iterations. It is expected that the AMT doubted between two (or multiple) documents and starting reading them before identifying it as correct. The outliers in the case of pseudo-relevance feedback are expected to be due to the fact multiple iterations were necessary (two and three iterations).

Parameter	PDF 1A	PDF 1B	BM25 1A	BM25 1B	Feedback 1A	Feedback 1B
AMTs that found correct document (out of 17)	13	14	16	16	16	16
Accuracy	75.5%	82.4%	94.1%	94.1%	94.1%	94.1%
Average time to find correct document (in seconds)	214	241	72	51	37	46
Average number of iterations	-	-	1.2	1.3	1.1	1.3
Average place of correct document in top 10 of results	-	-	4.1	5.7	3.9	6.3

Table 6.1: Results per systems for problem 1A and problem 1B separated.

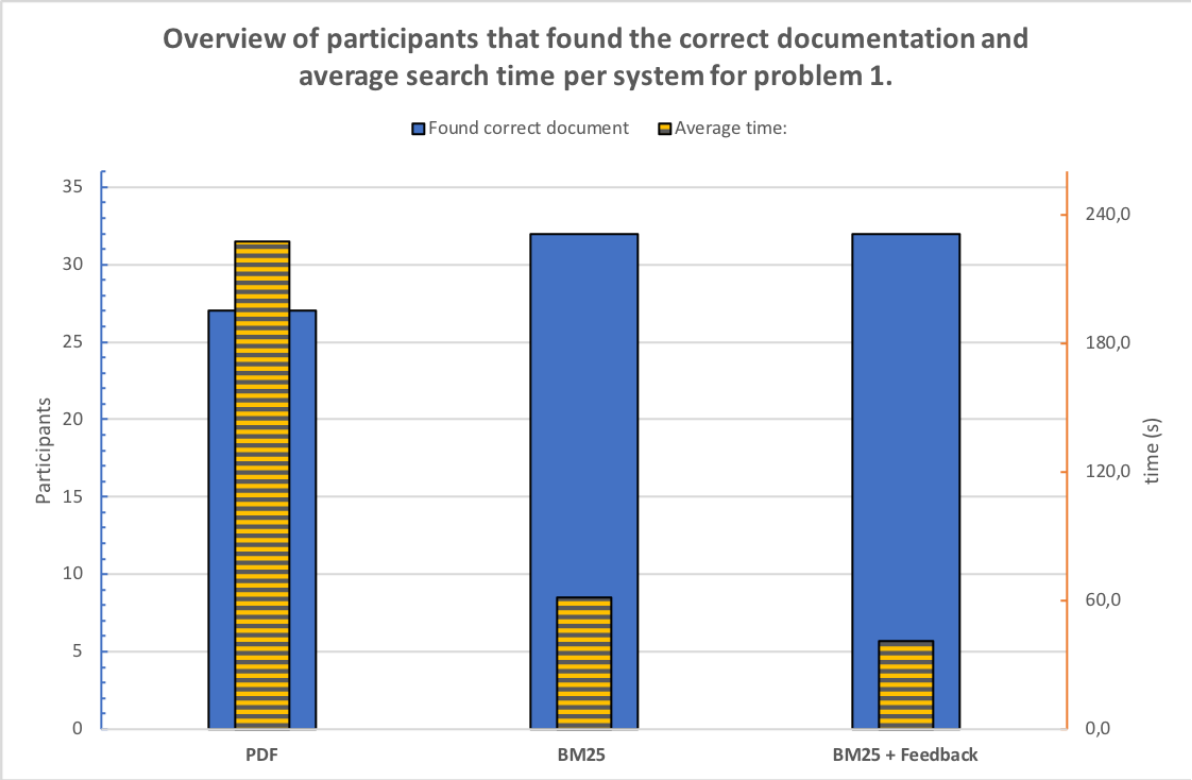


Figure 6.1: Number of AMTs that found the correct document including average search time for problem 1.

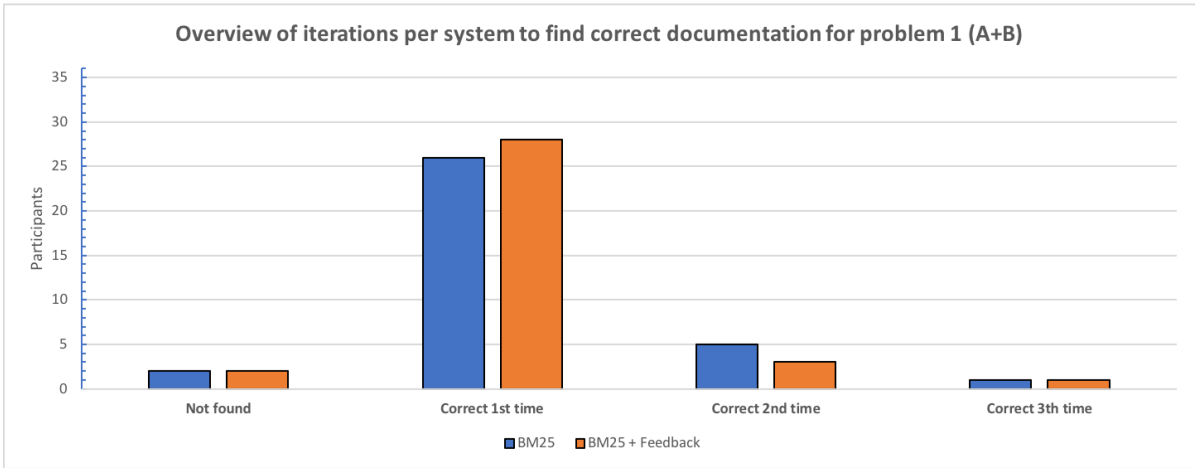


Figure 6.2: Overview of the number of iterations to find correct document for BM25 with and without feedback for problem 1.

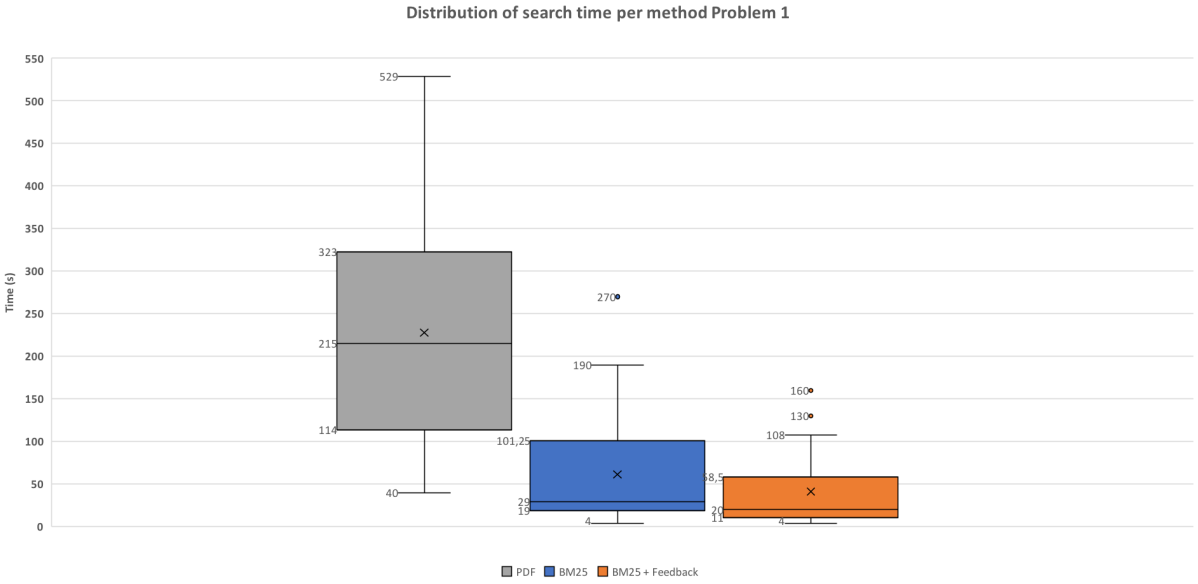


Figure 6.3: Boxplots of search time distribution per method for problem 1.

6.2. Problem 2

Table 6.2 shows an overview of the results from the second problem with a division in the A and B problem. Figure 6.4 shows the test results of problem 2 in total. 32 AMTs out of the 34 in total, were able to find the correct document using the PDF system, corresponding to 94% of the total AMTs, in an average search time of 217 seconds. The percentage of AMTs able to find the correct manual is high, but Figure 6.6 shows four outliers, where it took one AMT ten minutes to find this document.

The use of the BM25 algorithm results in 28 AMTs able to find the correct document in an average search time of 71 seconds. The same number of AMTs was able to find the correct document using the pseudo-relevance feedback, however, the average search time is decreased to 31 seconds. To be precise: the AMTs who were not able to find the document without feedback, were also not able to find it with feedback. This supports the assumption of pseudo-relevance feedback that the top results are assumed to be relevant, but is not always the case. Although the addition of pseudo-relevance feedback does not increase the number of AMTs able to find the correct document it is found the number of iterations necessary is lower for the feedback method. Twenty AMTs needed one iteration to find the document when the feedback method was enabled, against fifteen AMTs for the BM25 algorithm solely. Not only the number of iterations are less,

but Figure 6.6 shows the variation is search time is considerably lower for the method using feedback. The addition of feedback does result in faster retrieval with the same accuracy compared to the BM25 method for this problem.

Parameter	PDF 2A	PDF 2B	BM25 2A	BM25 2B	Feedback 2A	Feedback 2B
AMTs that found correct document (out of 17)	15	17	13	15	13	15
Accuracy	88.2%	100%	76.5%	88.2%	76.5%	88.2%
Average time to find correct document (in seconds)	245	193	119	30	41	22
Average number of iterations	-	-	1.9	1.3	1.5	1.2
Average place of correct document in top 10 of results	-	-	4.9	2.2	5.2	3.5

Table 6.2: Results per systems for problem 2A and problem 2B separated.

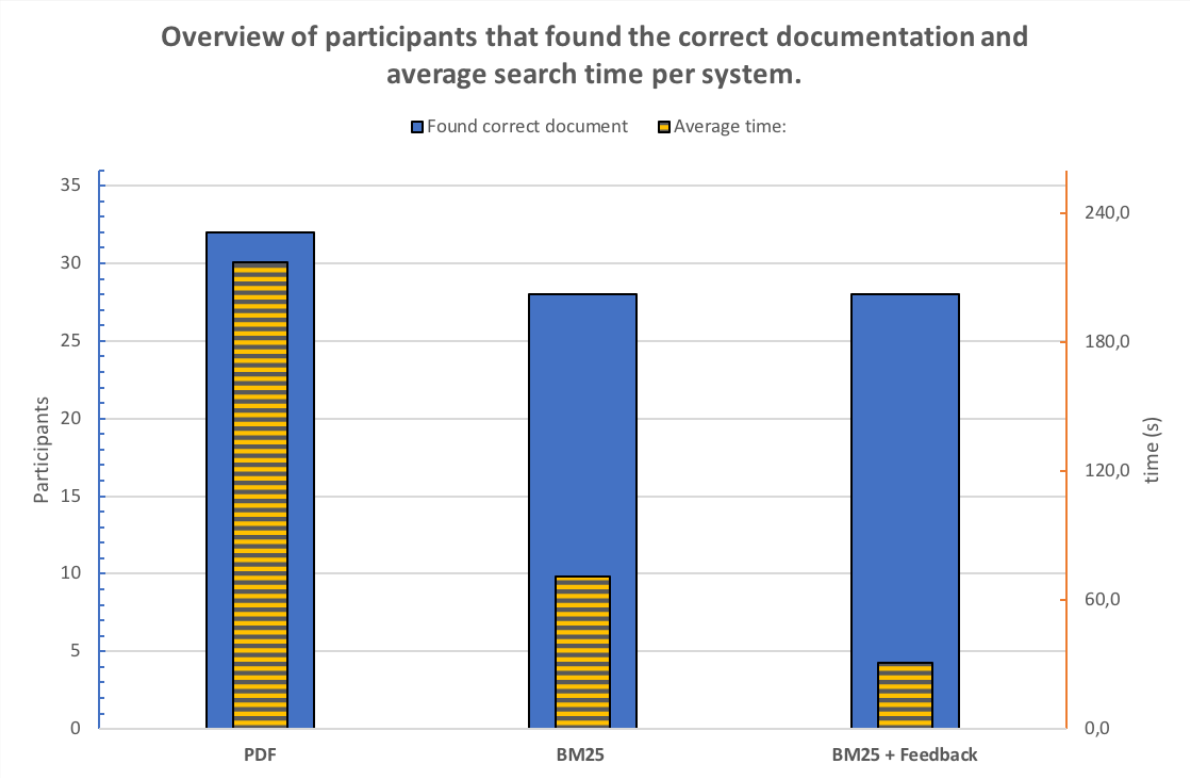


Figure 6.4: Number of AMTs that found correct document including average search time for problem 2.

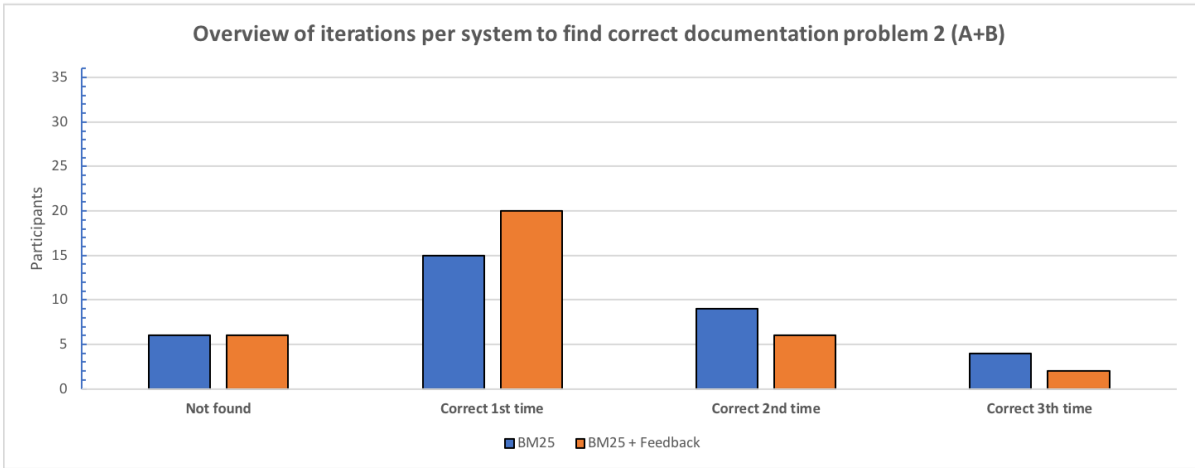


Figure 6.5: Overview of the number of iterations to find correct document for BM25 with and without feedback for problem 2.

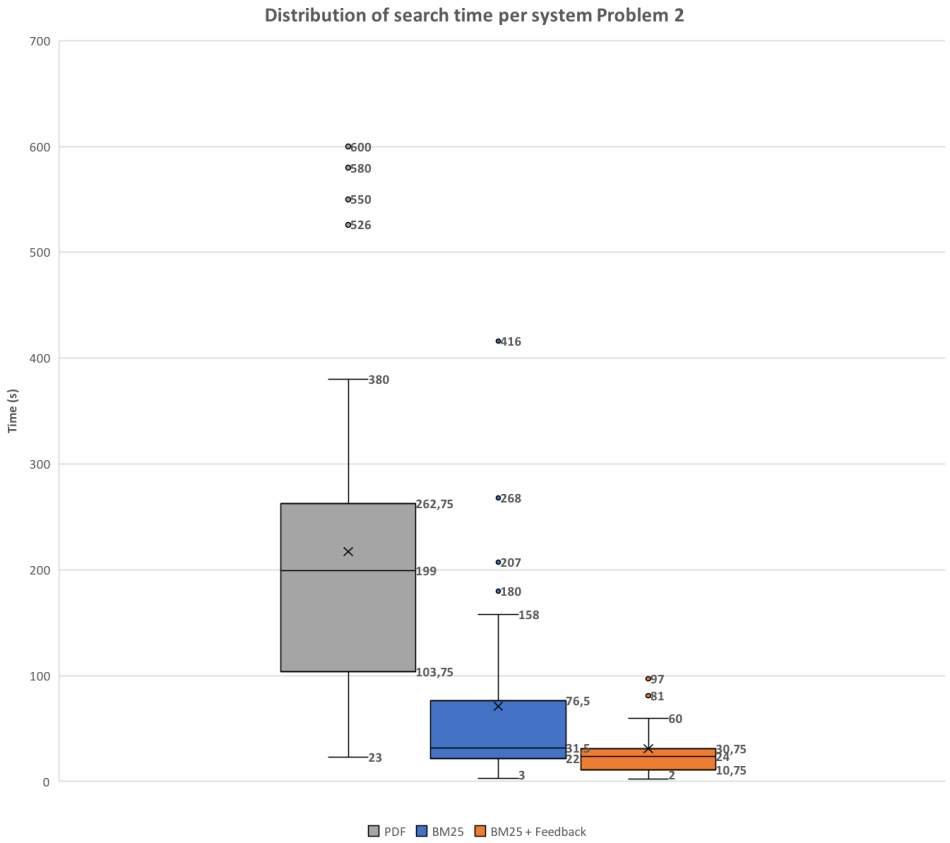


Figure 6.6: Boxplots of search time distribution per method for problem 2.

6.3. Problem 3

Table 6.3 shows an overview of the results from the second problem with a division in the A and B problem. The test results of problem 3 in total can be found in Figure 6.7. Remarkable is the difference between the results from the PDF search between problem A and B. Again, the difference in average search time using the PDF manuals is significantly higher compared to the other two methods. In total 30 AMT were able to find the correct document with an average search time of 165 seconds. Again, it costs an AMT ten minutes to find the correct document, and even one AMT (not shown in the figure) came after fifteen minutes to the conclusion that he was not able to find the correct manual.

The use of the BM25 algorithm results to an increase in number of AMTs able to find the correct document compared to the PDF method. Also the average search time is lowered to 34 seconds because most of the AMTs needed one query to find the correct AMM. This results in a small spread of search time for this method and a search time of two minutes is already identified as an outlier.

The use of the BM25 algorithm in combination with feedback does not increase the number of AMTs able to find the correct document. In fact, the same AMTs unable to find the correct document using the BM25 method, were also unable to find this document with the addition of pseudo-relevance feedback. The average search time is again lowered to 23 seconds, which cannot be explained by the number of iterations because these are equal (with and without feedback) and also the average place in the top results is higher for the method using feedback. The search time is stable since the spread is small and is slightly right skewed, meaning the average search time is low and only a couple AMTs needed some longer time.

Parameter	PDF 3A	PDF 3B	BM25 3A	BM25 3B	Feedback 3A	Feedback 3B
AMTs that found correct document (out of 17)	17	13	16	15	16	15
Accuracy	100%	76.5%	94.1%	88.2%	94.1%	88.2%
Average time to find correct document (in seconds)	187	138	35	34	25	21
Average number of iterations	-	-	1.1	1.2	1.1	1.2
Average place of correct document in top 10 of results	-	-	2.3	3.5	3.8	3.9

Table 6.3: Results per systems for problem 3A and problem 3B separated.

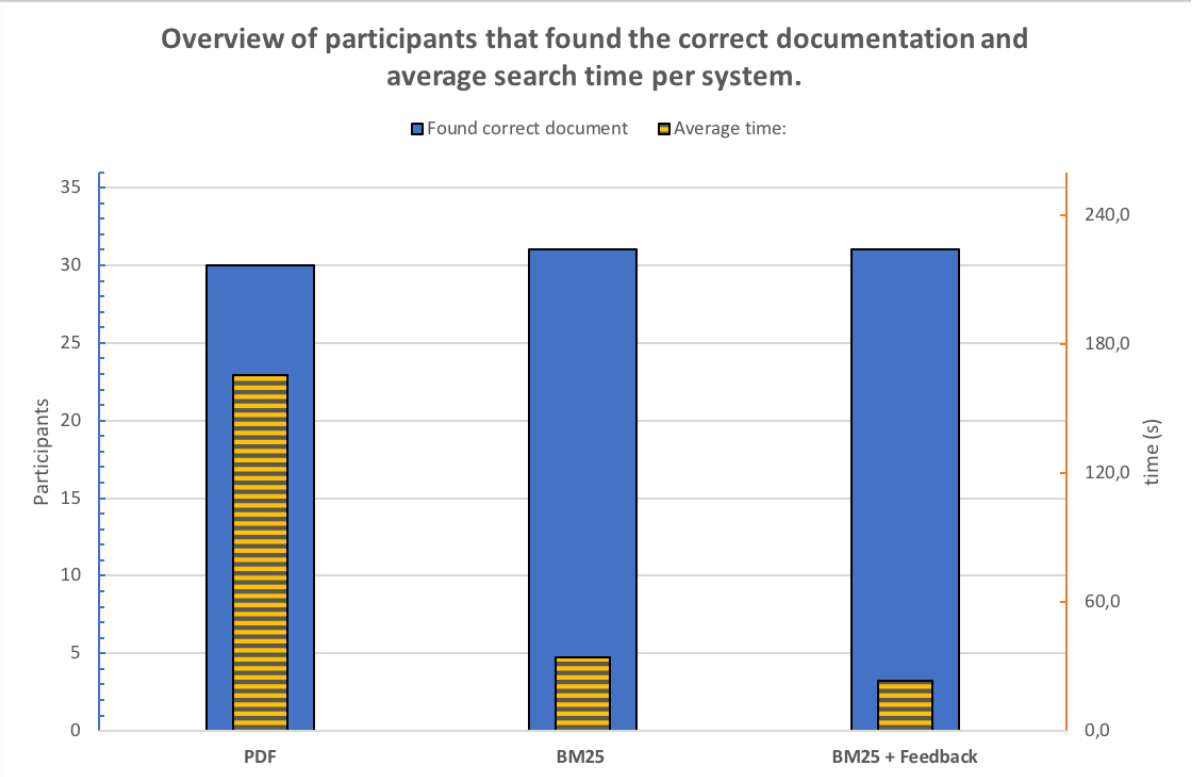


Figure 6.7: Number of AMTs that found correct document including average search time for problem 3.

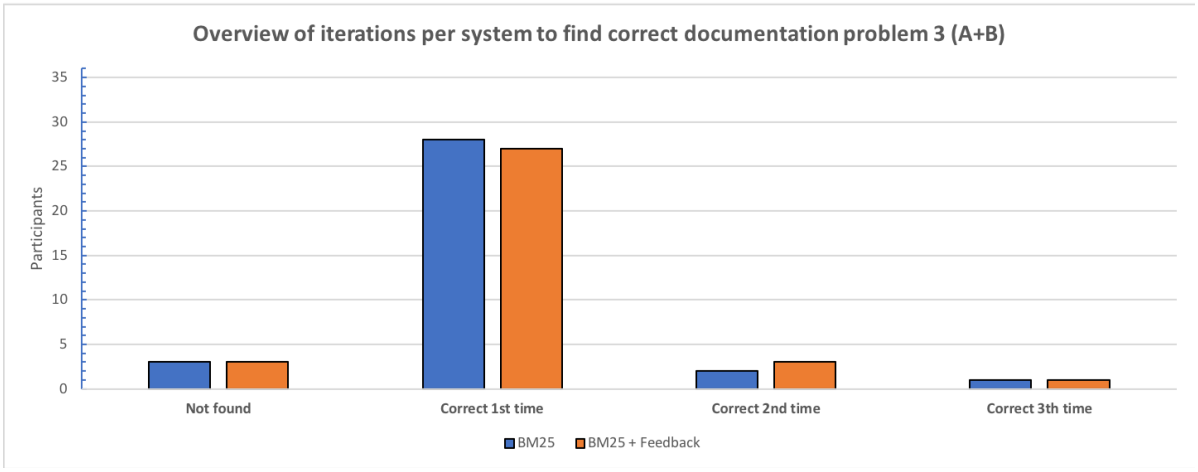


Figure 6.8: Overview of the number of iterations to find correct document for BM25 with and without feedback for problem 3.

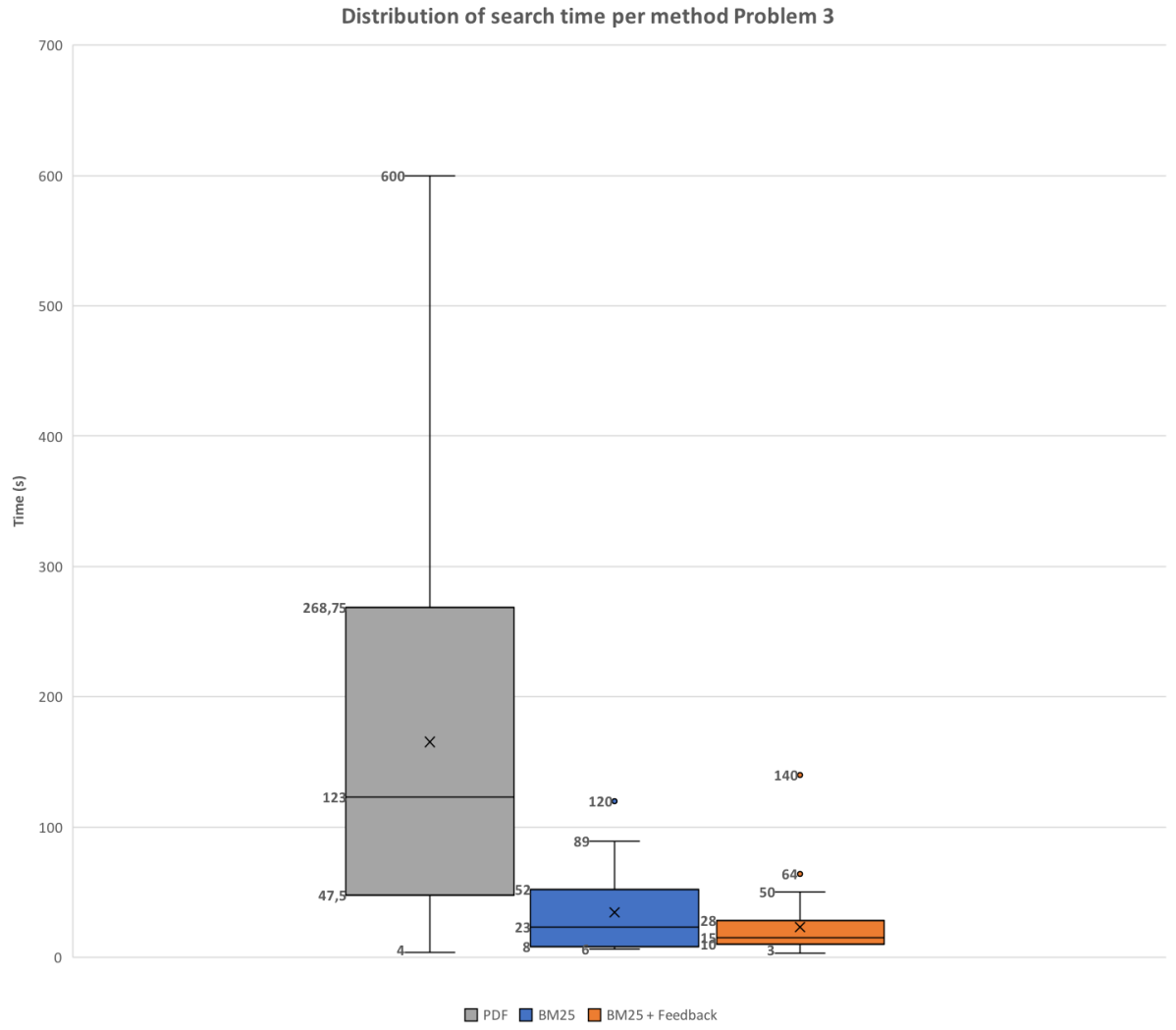


Figure 6.9: Boxplots of search time distribution per method for problem 3.

6.4. System Results

The results of all tests are bundled together per method, to make a system overview. A total of six tests is not enough to fully represent the system behaviour it gives a good first impression. The results from all tests per method are shown in Table 6.4. As one can see the percentage of AMTs able to find the correct document is comparable for all systems. One must be aware no end time was given for the current way of working with PDF documents, while for the other systems only three iterations were allowed and only documents from the top ten retrieved documents should be chosen. The results from Figure 6.10 show the spread in average search time for a PDF search is significant and over 25% of the AMTs need at least 4.5 minutes to find the correct documentation. This search time can range to ten minutes where one has to keep in mind this is only the time to find the documentation and there was no pressure on the AMT to finish the task on time.

By using the BM25 algorithm the average search time is lowered to 55 seconds, corresponding to a decrease of 72.9% compared to the PDF method. This is possible because 67.6% of the AMTs was able to find the correct manual after one iteration. The addition of feedback does not lead to more AMTs able to find the correct document in the top 10 of results, something which was already visible from the individual problem results. Therefore, it cannot be said the addition of feedback lead to more relevant results in general. A closer look shows the feedback method is able to retrieve more relevant results after one iteration, which is a possible reason why the feedback method is able to retrieve the relevant results faster. The average search time, compared to the BM25 method, to find the correct document is decreased by 42% due to the addition of feedback. Interesting is the average place where the correct document is found is not increased for the feedback procedure, which might be the result of a decrease in the number of iterations. The correct document is possibly found earlier when feedback is enabled, because it appeared (higher) in the top 10 of results, while without feedback the document was not in the top 10 of results. Therefore an extra iteration was not necessary for the feedback procedure. For the BM25 this extra iteration made it possible for the correct document to show up (higher) in the top 10 of results. No conclusive evidence for this is found and more research is necessary to ground this statement.

Parameter	**PDF**	**BM25**	**BM25 + Feedback**
AMTs that found correct document (out of 102)	89	91	91
Percentage AMTs that found correct document	87.3%	89.2%	89.2%
Percentage AMTs needed one iteration	-	67.6%	73.5%
Percentage AMTs needed two iterations	-	15.7%	11.8%
Percentage AMTs needed three iterations	-	5.9%	3.9%
Average time to find correct document (in seconds)	203	55	32
Average number of iterations	-	1.31	1.22
Average place of correct document in top 10 of results	-	3.8	4.4

Table 6.4: Results per method for all tests.

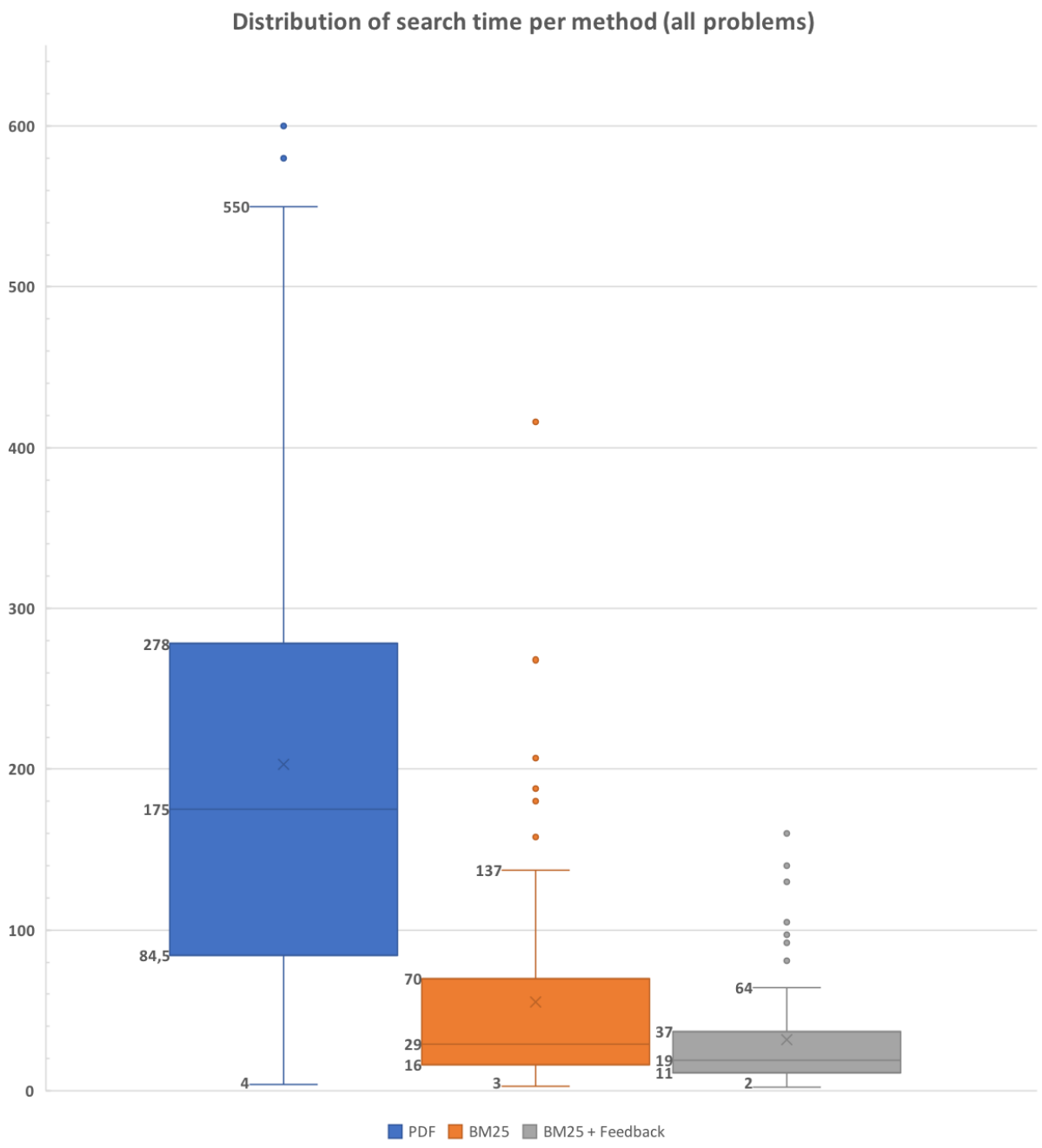


Figure 6.10: Boxplots of search time distribution per method.

6.5. Validation

To validate the results from the tests a validation test must be conducted. To validate if the lower average search time is due to the addition of feedback it is required to conduct an additional statistical test. The sample size is not in all of the test 30 or above, and therefore one must be careful before standard tests are applied. The boxplots, of both the BM25 method without and including feedback from the separate results, shows the data is not normally distributed (skeweness of the data). To be independent of the data distribution, and not to rely on assumptions that data for a population is following a particular (e.g. normally) distribution, non-parametric statistics are applied [8]. The sample size for all tests are relatively small, are matched pairs because they have a before and after situation (without and with feedback), make use of one group of participants and are measured on a continuous scale. A non-parametric test satisfying all criteria is the Wilcoxon signed-rank test, which compares the means of the samples.

The Wilcoxon Signed-Rank test is a non-parametric test for two dependent samples and is known as the alternative for the paired t-test. The Wilcoxon Signed-Rank test is used when some of the assumptions of the t-test are not met. In case of the given results this is either that the data is not normally distributed, or the sample size (of 30) is not met. The Wilcoxon Signed-Rank test is a hypothesis test that attempts to make a claim about the difference in score from paired samples for the population median. The null hypothesis is the statement that a difference in the median is due to chance while the alternative hypothesis claims the opposite. The main properties for this test, making use of two paired samples, are:

- Two dependent samples. Paired or matched samples or repeated measures (measures taken from the same subjects).
- The test is a non-parametric test and does not require the normality assumption.
- Data must be a least at ordinal scale, so it can be ranked.

To make use of the Wilcoxon Signed-Rank test, one first computes the difference, the absolute difference, and the sign of the difference between the samples. Next, the samples which are tied (same value) are removed from the data and the samples are organized according the absolute differences in ascending order. The samples are ranked, where samples with the same absolute value difference receive the same rank.

The formula for the statistic for the Wilcoxon's Signed-Rank test is given by Equation 6.1:

$$T = min[W^+, W^-] \tag{6.1}$$

Where W^+ is the sum of the positive ranks, and W^- the sum of the negative ranks. When the number of pairs is large (used for the results per method), the normal distribution can be used. From here the z statistic can be calculated, shown in Equation 6.2:

$$z = \frac{T - \frac{n(n+1))}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}} \tag{6.2}$$

Where n is the number of pairs.

The following null hypothesis and alternative hypothesis are tested for a significance level of $\alpha = 0.05$ and a two-tailed type.

H_0: Median (difference) = 0

H_a: Median (difference) $\neq$ 0

The calculated T values are compared to the critical T value (T_{crit}). This T_{crit} can be found in a table with critical values of the wilcoxon signed rank test (given in Appendix B). When T $\leq$ T_{crit} the null hypothesis is rejected. From the results a correlation coefficient r can be calculated as a measure of the effect size, shown in Equation 6.3:

$$r = \frac{z}{\sqrt{2n}} \tag{6.3}$$

The effect size provides an objective measure of the effect of the addition of pseudo-relevance feedback [8], where:

- r = 0.10 - 0.30: small effect
- r = 0.30 - 0.5: medium effect
- r = 0.5 - 1.0: large effect

Problem 1

For this Wilcoxon signed-rank test the null hypothesis H_0: median (difference) = 0, i.e. any difference between the BM25 algorithm and the BM25 algorithm + feedback is due to chance and the alternative hypothesis is H_1: median (difference) $\neq$ 0 is tested with a significance level of $\alpha = 0.05$.

To be significant, the calculated T value must be equal or less than the critical T value. After conducting the Wilcoxon signed-rank calculation it is found T = 151 and T_{crit} = 116. So T = 151 > 116 = T_{crit} and therefore the result is not significant and the null hypothesis is accepted. There is more than 5% probability the results are due to chance, meaning there is not significant evidence the feedback methods results in better search times to solve problem 1. The results from the Wilcoxon signed-rank test are summarised in Table 6.5

Parameter	**Value**
n	28
T	151
Mean	203
Variance	1928.5
Standard deviation	43.91
z-score	1.184
T_{crit}	116
p-value	0.236
Significant?	Not
Decision	Accept H_0
r	0.224

Table 6.5: Results from two-tailed Wilcoxon signed-rank test for problem 1 with significance level 0.05.

Conclusion: there is not sufficient evidence to suggest there is any difference between the BM25 algorithm and BM25 algorithm + feedback in terms of search time due to the addition of feedback. Meaning it is not proven that the addition of pseudo-relevance feedback results in a faster retrieval of the relevant document.

Problem 2

For this Wilcoxon signed-rank test the null hypothesis H_0: median (difference) = 0, i.e. any difference between the BM25 algorithm and the BM25 algorithm + feedback is due to chance and the alternative hypothesis is H_1: median (difference) $\neq$ 0 is tested with a significance level of $\alpha = 0.05$.

To be significant the calculated T value must be equal or less than the critical T value. After conducting the Wilcoxon signed-rank calculation it is found T = 83 and T_{crit} = 89. So T = 83 < 89 = T_{crit} and therefore the result is significant and the null hypothesis is rejected. There is less than 5% probability the results are due to chance, meaning there is significant evidence that the feedback methods results in better search times to solve problem 2. The results from the Wilcoxon signed-rank test are summarised in Table 6.6

Parameter	Value
n	25
T	83
Mean	162.5
Variance	1381.25
Standard deviation	37.17
z-score	2.139
T_{crit}	89
p-value	0.032
Significant?	Yes
Decision	Reject H_0
r	0.428

Table 6.6: Results from two-tailed Wilcoxon signed-rank test for problem 2 with significance level 0.05.

Conclusion: there is sufficient evidence to suggest there is a positive difference between the BM25 algorithm and BM25 algorithm + feedback in terms of search time due to the addition of feedback. Meaning the addition of pseudo-relevance feedback results in a faster retrieval of the relevant document.

Problem 3

For this Wilcoxon signed-rank test the null hypothesis H_0: median (difference) = 0, i.e. any difference between the BM25 algorithm and the BM25 algorithm + feedback is due to chance and the alternative hypothesis is H_1: median (difference) $\neq 0$ is tested with a significance level of $\alpha = 0.05$.

To be significant the calculated T value must be equal or less than the critical T value. After conducting the Wilcoxon signed-rank calculation it is found T = 93 and T_{crit} = 107. So T = 93 < 107 = T_{crit} and therefore the result is significant and the null hypothesis is rejected. There is less than 5% probability the results are due to chance, meaning there is significant evidence that the feedback methods results in better search times to solve problem 3. The results from the Wilcoxon signed-rank test are summarised in Table 6.7

Parameter	Value
n	27
T	93
Mean	189
Variance	1732.5
Standard deviation	41.62
z-score	2.318
T_{crit}	107
p-value	0.020
Significant?	Yes
Decision	Reject H_0
r	0.446

Table 6.7: Results from two-tailed Wilcoxon signed-rank test for problem 3 with significance level 0.05.

Conclusion: there is sufficient evidence to suggest there is a positive difference between the BM25 algorithm and BM25 algorithm + feedback in terms of search time due to the addition of feedback. Meaning the addition of pseudo-relevance feedback results in a faster retrieval of the relevant document.

Methods

In Table 6.8 and Figure 6.10 the results are shown to check whether the assumption of non-normally distributed data was correct. In fact, if the data is normally distributed other more powerful tests can be applied since the sample size is sufficient. From the boxplot it is clear none of the systems produces normally distributed data (both are highly skewed and not symmetrical), indicating the use of the non-parametric statistical tests is best to apply. This is also confirmed by the calculation of the skewness and excess kurtosis. A value

for the skewness above +1 indicates highly positively skewed data which is for both methods the case. The values for the kurtosis are above 3 meaning they can be named leptokurtic (heavy tailed). For both methods most of the AMTs were able to find the documentation in a search time lower as the average for that method.

Parameter	**PDF**	**BM25**	**BM25 + Feedback**
Sample size (n)	89	91	91
Sample average ($\overline{x}$)	203.0	55.2	31.9
Sample standard deviation	145.0	68.8	33.9
Skewness	0.97	2.68	1.99
Excess kurtosis	0.58	9.04	3.54

Table 6.8: Results of normality check per system.

Parameter	**Value**
n	80
T	944
Mean	1620
Variance	43470
Standard deviation	208.5
z-score	3.242
T_{crit}	1211
p-value	0.001185
Significant?	Yes
Decision	Reject H_0
r	0.362

Table 6.9: Results from two-tailed Wilcoxon signed-rank test for all results with significance level 0.05.

For the Wilcoxon signed-rank test of all data the null hypothesis H_0: median (difference) = 0, i.e. any difference between the BM25 algorithm and the BM25 algorithm + feedback is due to chance and the alternative hypothesis is H_1: median (difference) $\neq 0$ is tested with a significance level of $\alpha = 0.05$.

The T value equals 944 and $T_{crit} = 1211$ and because $T < T_{crit}$ the the null hypothesis H_0 can be rejected. In other words, the difference between the values of the BM25 algorithm and the BM25 algorithm + feedback is large enough to be statistically significant. A summary of the results can be find in Table 6.9.

Conclusion: there is sufficient evidence to suggest there is a positive difference between the BM25 algorithm and BM25 algorithm + feedback in terms of search time due to the addition of feedback. Meaning the addition of pseudo-relevance feedback results in a faster retrieval of relevant documents.

7

Conclusions and Recommendations

This final chapter gives the concluding remarks, discusses the research contribution, and recommendations for future research are proposed.

7.1. Conclusions

The execution of aircraft maintenance has always been a time-consuming task and with airliners increasing the productivity of aircraft, the pressure on AMTs is high. The high pressure on AMTs is a well-known problem in the industry. The combination of high pressure on the AMT and the enormous amount of maintenance documents, which are either paper-based or almost unsearchable, puts the AMT for a terrible decision. Either save some time by not consulting documentation or risk the chance on flight delays. This impossible situation was the reason to make this the problem statement of this thesis: **"Aircraft maintenance technicians are unable to find the correct maintenance documentation and perform their corresponding tasks during aircraft turnaround time."**.

It is decided to focus on the improvement of finding relevant digital maintenance documentation as a solution to the problem statement. The paper on-screen solutions are the first step in this digitalisation, but also come with some limitations because they are, for example, hard to search through and only exact keyword matches will deliver results. Supported by the fact that, the transition from paper-based documentation towards digital documentation is in the early stages, it is important to have a reference of what is possible with ranking algorithms in the future of aircraft maintenance. Researchers already spent much effort on document retrieval in the IR domain and from here it is found the BM25 algorithm is used very often as a consistently good performing algorithm. As the literature on maintenance systems in the aircraft domain is limited, it is unknown if ranking aircraft documentation delivers the same results compared to other domains. Many algorithms could be investigated, but without a solid baseline, it is difficult to compare different methods in the future. It is due to this gap in literature the first research question is stated as:

"Does the implementation of the BM25 algorithm lead to more relevant search results for aircraft maintenance technicians searching for documentation, and what is the time efficiency impact due to this algorithm?"

It was expected that more experience and knowledge would lead to better results in a shorter time for the BM25 algorithm because the AMT would be able to use more specific queries resulting in fewer iterations, with the correct document higher in the top of results.

The outcome of the experiments showed mixed results for the three categories of problem definitions as formulated in Table 4.1, which have decreasing frequency in daily operations. The first two problems (1A and 1B), which the AMT is most likely to face, showed the accuracy of the BM25 algorithm was higher compared to the current way of working with PDF documentation. From the total of 34 AMTs, only 27 were able to find the correct document using the PDF document while 32 were able to find it making use of the BM25 algorithm. Not only the accuracy is increased due to the use of the algorithm, also the time efficiency is improved. Going

from 214 seconds on average to 72 seconds for problem 1A and from 241 seconds to 51 seconds for problem 1B can be seen as an improvement.

The most surprising results come from the data of the problems in the second category. In none of the two problems, the use of the BM25 algorithm showed any improvement in accuracy. The number of AMTs able to find the correct document is dropped from 32 (PDF) to 28. Surprising is the time difference between problem A and B for the BM25 algorithm results (119 seconds against 30 seconds on average). A possible explanation for this is can be found in the task description of problem 2B, the exact words are already described, while this is not the case for problem 2A. This is supported by the fact the average number of iterations is significantly lower (1.3 vs 1.9) and the average place in the top 10 significantly higher (2.2 vs 4.9). These results support the hypothesis that more knowledge leads to faster retrieving of the correct documentation. These problems are less frequently occurring as the problems from the first case, which can also be seen in the average search time. However, one must be aware the range in average search time between the A and B problem is large.

The problem cases in the third category showed that, in general, the accuracy of the BM25 algorithm is higher compared to the current way of working. However, the difference between the AMTs able to find the correct document using the algorithm and PDF documentation was only one person (30 and 31). The difference in results between problem 3A and 3B solved by PDF documents is remarkable. For problem 3A 100% of the AMTs was able to find the correct document, while for problem 3B only 76.5% of the AMTs was able to find it. A possible reason for this could be that although problem 3A does not occur many of times in practice, it is a simple to imaging problem case and solution. The AMT might, therefore, had a good idea in which section of the chapter he had to look to perform the task. Problem 3B, however, is very specific and advanced and might therefore not be found as often. To find out the exact reason it is best to perform the test again and ask the AMTs for a reason why they did not find the task. Although the accuracy of the BM25 algorithm in this problem is not significantly higher, the time difference is significant. The results conflict with the mentioned hypothesis because these problems do occur the less frequent, but AMTs were able to find them in the shortest time (for all methods).

The results are taken together, to provide results per method, and from this, the first research questions can be answered. In general, it is concluded that the implementation of the BM25 algorithm does not lead to more relevant results but performs at least as good as the current way of working with PDF manuals. In 50% of the problem cases it gave better results, while in 50% it did not. An important note in here is that for the use of the BM25 algorithm a limit of three iterations was given to the AMTs and only the top 10 of results was available to investigate. Although this might have an influence on the number of AMTs able to find the correct document, it was a necessary step in avoiding AMTs starting a manual search (and thereby not making use of the ranking algorithm). Still, in 89.2% of the cases, the AMTs were able to find the correct document by using the BM25 algorithm. On average, it took an AMT 55 seconds and 1.3 iterations (this means slightly more than one iteration on average per person) to find the correct document. This average time to find the correct documentation corresponds to only 27% of the original time when the PDF manuals were used.

The use of relevance feedback has shown some potential in literature because the technique is able to improve the users' initial query and facilitate a better document retrieval. However, like the BM25 algorithm, no information is found for the aircraft maintenance domain and therefore the second research question is defined as:

Does the addition of pseudo-relevance feedback to the BM25 algorithm lead to more relevant search results compared to the BM25 alone, and what is the time efficiency impact due to feedback?

The results show the addition of pseudo-relevance feedback did not result in the retrieval of more relevant documents in any of the problems compared to the BM25 algorithm. The correlation between feedback and time efficiency is interesting because the Wilcoxon signed-rank test showed in two of the three cases it is not by chance results are retrieved faster. Compared to the current way of working with PDF documents, there is an average decrease in search time of almost three minutes (171 seconds). This corresponds to a decrease of 42% compared to the method where solely the BM25 algorithm is used. This is an unexpected result because the results of the sensitivity analysis showed in only one of the four cases the feedback mechanism performed better compared to the BM25 algorithm on its own. The sensitivity analysis also showed the best results are retrieved when only one document and one file are assumed to be relevant for the pseudo-relevance feedback. However, during the experiment, the top five documents and the top ten words were chosen to be relevant (because the sensitivity analysis was performed after the experiment due to time constraints). It in-

dicates better results are possible, but one must be aware the test size of the sensitivity analysis was low.

Concluding on this research study into implementing the BM25 algorithm in the aircraft maintenance domain is that it has been shown that it does provide significantly faster results as the current method making use of PDF manuals. This study touched upon the possibilities which ranking algorithms can offer to the aircraft maintenance domain. In this study, it is shown an AMT spent 203 seconds on average to find the correct document using PDF documentation. The average search time is lowered to 55 seconds by using the BM25 algorithm as search method. The addition of pseudo-relevance feedback to the BM25 algorithm made it possible to lower the average search time to 32 seconds, which is 15.8% of the original search time (PDF). These results, while preliminary, suggest it is likely that AMTs who only spend 15.8% of their original search time on finding documentation, have to make the aforementioned trade-off, between time and safety, much less. It is advised to academia and the industry to invest more in research for ranking functionalities in the aircraft maintenance domain because one algorithm is able to significantly reduce the AMTs search time.

The use of ranking algorithms, and in particular the BM25 algorithm, is not new but the adoption in the aircraft maintenance domain is limited. This thesis is (one of) the first that investigated the impact of the BM25 algorithm on accuracy and efficiency. It is able to retrieve the same accuracy, but with higher efficiency, as the current way of working. From here the baseline is set for new research in the aircraft maintenance domain to investigate if accuracy, as well as efficiency, can be increased. Literature already provides many algorithms or variations of the BM25 algorithm in other domains able to retrieve an even higher efficiency. Notwithstanding the limitations, this thesis suggests the possible efficiency increase of using ranking algorithms in the aircraft maintenance domain must not be underestimated. Much effort is spent on more efficient repair methods, while for efficiency improvements, the industry might need to consider to focus on more efficient search methods.

7.2. Recommendations

The research has been one of the first attempts to thoroughly examine the use of the BM25 algorithm is a valuable addition for aircraft maintenance in terms of efficiency. The addition of pseudo-relevance feedback has shown efficiency is increased further without losing accuracy. Although considered as valuable already, there is always room for improvement. As mentioned already in Chapter 3 the research is bounded by a scope and other limitations. To researchers interested in future development of ranking algorithms in the aircraft maintenance domain, or to further extend the current model, several recommendations are given.

General

The first general limitation is that the AMTs were still in training. It is without a doubt this group has less experience compared to AMTs in the field. Although more participants were able to perform in the experiment, the total number of different participants is low. If possible, it is recommended to conduct new research by using experienced AMTs to investigate whether the effect of experience is significant or not. As the results indicate more experience does not always lead to the faster retrieval of relevant documents, it might be questioned if this result was due to the manner of asking the questions. It is possible the faster retrieval of the results is due to the fact a word was present in both the question as well as in the name of the AMM. An interesting question is if the same results are present when the problem cases are tested on the tarmac without asking the question to the AMT on paper.

Also recommended is to investigate if the implementation of a table of contents per chapter leads to a higher accuracy of retrieved documents. During the experiment, many AMTs went to the table of contents (for the PDF method) to find out which specific section to use. The addition of this table of contents might become handy when only general search results are found for specific queries. By a simple questionnaire it is possible to test if this table of contents is also often used by experienced AMTs. Indicated multiple times is a more stable and computationally faster server is a must for the industry. The server must be able to handle more users at the same time and perform the ranking calculations faster. For research purposes, it is preferable to have a more powerful server, although it is not a must if the number of users is below ten people.

Model

A natural progression of this work is to optimise the model. First, the Python code itself can be improved or to make it consistent with Macster; the programming language can be changed to Javascript. Secondly, the functionalities of the model can be expanded. At the moment the model is only able to search for text, while AMTs also want to use technical drawings. It is also recommended to implement a search option able to search for figures and drawings, with an option to specifically filter on it. One must be aware that different ranking algorithms are used to retrieve pictures and drawings. The functionality of the filter function can further be extended to filter on tail number. One of the advantages is an AMT cannot perform a task with a manual corresponding to a different aircraft type. Also by limiting the number of documents, the necessary calculations for the ranking algorithm require less powerful servers or calculations are performed faster. This thesis focussed on the implementation of ranking algorithms, but no time is spent on user-friendliness. The search bar is a simple option to find documentation, but many improvements to make the model more user-friendly are possible. One can implement visuals (bars, scores, or percentages) that show the user how well the retrieved documents relate to their query, while at the moment the user does not know this.

Ranking

The literature study showed many different ranking algorithms are used in various research fields. Researchers in other domains as aircraft maintenance have already demonstrated other ranking algorithms can be more efficient than the BM25 algorithm. In general, it is therefore recommended to test whether this statement also holds for the aircraft maintenance domain. It is best to use the same collection of documents, as used in this study, to fairly compare the results. To be more specific, it is also possible to extend the BM25 ranking algorithm making use of different weights (k_1 and b) for different fields, e.g., title and tables. This variant of the BM25 algorithm is called BM25F. Another option is to use the BM25+ variant which uses one additional parameter, to let long documents which match a query score more fairly. Another possibility is to develop a library of technical words in the aircraft maintenance domain that makes stemming possible. Stemming reduces or derives words to their root and therefore it is no longer necessary to have exact word matches. This can be supplemented with lemmatisation which makes it possible to distinguish the meaning of a word (when one word has several contexts). By developing a technical word library, it is also possible to compare the most frequently used words with this library. When the words are not in the library but very commonly used, these so-called "stop words", can be removed from the calculations in the algorithm to make a retrieval performance improvement possible. Another improvement to the ranking algorithm is to implement a function that compares the order in which the words occur. The current model does not take this word order into account while this might have a significant influence on the meaning of the user.

The sensitivity analysis showed for low k_1 and b values the best results are retrieved. This is remarkable because it suggests the IDF formula would lead to better retrieval results compared than the more complex BM25 algorithm and therefore it is recommended to investigate this in future research. Also, the number of documents has a high impact on the results. A possible extension of this research is to continue with the model and extend the number of maintenance manuals and do a thorough investigation of its impact. This can be done in combination with the research if the IDF algorithm leads to more relevant and faster results because only some parameters in the model have to be changed.

Feedback

During the experiment the k_1, b, N and M values where fixed. More research is necessary to investigate the impact of changing these values. Before this experiment, it was hard to perform a better evaluation because it was unknown which queries the AMTs would use. Although the queries are not shown in this report, they are available on paper. Also interesting is to take into account the results from the sensitivity analysis. Although the Wilcoxon signed-rank test shows the feedback results are significant for the system as a whole, it is suggested to perform an extensive sensitivity analysis for the pseudo-relevance feedback mechanism on aircraft maintenance documents. This because the results from the feedback sensitivity analysis showed no evidence and is in contradiction to the results of the experiment.

To be less dependent on the user input and to lower the amount of data storage, there is chosen to make use of pseudo-relevance feedback. In case there are more AMTs available for a longer period, one must consider the option to implement other feedback methods. In particular implicit feedback options might be

useful because the AMT does not notice the use of it, but many data can be retrieved from the model and stored. By applying machine learning techniques, this data can be used to make a prediction on which aircraft maintenance manual is requested by the AMT. Another option is to create a feedback loop able to adjust the k_1, b, N, and M after each iteration. By using a feedback loop in this way, the model can adapt itself and possibly find a global optimum (for different query).

Bibliography

[1] RC Baldwin. How do you spell e-maintenance?, 2004. URL http://www.mt-online.com. Date Accessed: 14-04-2018.

[2] Lorenz Bernauer, Eun Jin Han, and So Young Sohn. Term discrimination for text search tasks derived from negative binomial distribution. *Information Processing & Management*, 54(3):370 – 379, 2018. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2018.01.003. URL http://www.sciencedirect.com/science/article/pii/S0306457318300207.

[3] Olov Candell, Ramin Karim, and Peter Söderholm. emaintenance-information logistics for maintenance support. *Robot. Comput.-Integr. Manuf.*, 25(6):937–944, December 2009. ISSN 0736-5845. doi: 10.1016/j.rcim.2009.04.005.

[4] Capgemini. Smart Factories to add 500 billion to the global economy in next 5 years. Technical report, Capgemini, 2017. URL https://www.capgemini.com/wp-content/uploads/2017/05/dti-smartfactories-webversion.pdf.

[5] Shane Connelly. Practical bm25 - part 1: How shards affect relevance scoring in elasticsearch, 04 2018. URL https://www.elastic.co/blog/practical-bm25-part-1-how-shards-affect-relevance-scoring-in-elasticsearch. Date Accessed: 19-05-2018.

[6] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company, USA, 1st edition, 2009. ISBN 0136072240, 9780136072249.

[7] Hiemstra D. Information retrieval models. *Information Retrieval: searching in the 21st Century*, pages 2–19, 11 2009.

[8] Andy Field. *Discovering Statistics Using IBM SPSS Statistics*. Sage Publications Ltd., 4th edition, 2013. ISBN 1446249182, 9781446249185.

[9] Apache Software Foundation. Powerful, accurate and efficient search algorithms, 2018. URL https://lucene.apache.org/core/. Date Accessed: 21-05-2018.

[10] Ben Hargreaves. Industry 4.0's Impact on Aviation Design And MRO, 2017. URL http://aviationweek.com/connected-aerospace/industry-40-s-impact-aviation-design-and-mro. Date Accessed: 18-03-2018.

[11] P. Hu, S. Ye, L. Yu, and K. R. Lai. Valence-arousal analysis for mental-health document retrieval. In *2017 International Conference on Orange Technologies (ICOT)*, pages 61–64, Dec 2017. doi: 10.1109/ICOT.2017.8336089.

[12] IATA. Airline maintenance cost executive summary. An Exclusive Benchmark Analysis (FY2014 data) by IATA's Maintenance Cost Task Force. Technical report, IATA, 2015. URL https://www.iata.org/whatwedo/workgroups/Documents/MCTF/AMC-Exec-Comment-FY14.pdf.

[13] IATA. Guidance material for the implementation of paperless aircraft operations in technical operations, November 2017. URL https://www.iata.org/whatwedo/ops-infra/Documents/Paperless%20Aircraft%20Operations%20in%20Technical%20Operations%20-%20Guidance%20Material%20for%20Implementation.pdf. Date Accessed: 20-08-2018.

[14] ICAO. Guidance for acceptance of electronic aircraft maintenance records (eamr), 11 2017. URL https://www.icao.int/safety/airnavigation/OPS/airworthiness/Pages/EAMR.aspx. Date Accessed: 28-08-2018.

[15] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 154–161, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. doi: 10.1145/1076034.1076063. URL http://doi.acm.org/10.1145/1076034.1076063.

[16] H.A. Kinnison and T. Siddiqui. *Aviation Maintenance Management, Second Edition*. McGraw-Hill Education, 2012. ISBN 9780071805032.

[17] H. Koornneef, W.J.C. Verhagen, and R. Curran. Contextualising aircraft maintenance documentation. *International Journal of Agile Systems and Management*, 10:160, 01 2017.

[18] Wessel Kraaij. Variations on language modeling for information retrieval. *SIGIR Forum*, 39(1):61–61, June 2005. ISSN 0163-5840. doi: 10.1145/1067268.1067291. URL http://doi.acm.org/10.1145/1067268.1067291.

[19] Matthias Lampe, Martin Strassner, and Elgar Fleisch. A ubiquitous computing environment for aircraft maintenance. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, pages 1586–1592, New York, NY, USA, 2004. ACM. ISBN 1-58113-812-1. doi: 10.1145/967900.968217. URL http://doi.acm.org/10.1145/967900.968217.

[20] S.g. Lee, Y.-S. Ma, G.l. Thimm, and J. Verstraeten. Product lifecycle management in aviation maintenance, repair and overhaul. *Computers in Industry*, 59(2-3):296–303, 2008. doi: 10.1016/j.compind.2007.06.022.

[21] Eric Levrat, Benoît Iung, and Adolfo Crespo Marquez. E-maintenance: Review and conceptual framework. *Production Planning & Control - PRODUCTION PLANNING CONTROL*, 19:408–429, 06 2008.

[22] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715. doi: 10.1017/CBO9780511809071.

[23] David J. Spanovich Paul Seidenman. Why Airlines, Aftermarket Struggle With Digital Record-Keeping, 2016. URL http://aviationweek.com/connected-aerospace/why-airlines-aftermarket-struggle-digital-record-keeping. Date Accessed: 18-04-2018.

[24] S.E. Robertson and K.S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976. doi: 10.1002/asi.4630270302.

[25] Lennart Scherp. Towards automated classification of aircraft maintenance documentation. Master's thesis, University of Technology Delft, 12 2016.

[26] Eileen Sobbe, Ralf Tenberg, and Hans Mayer. Knowledge work in aircraft maintenance. *Journal of Technical Education (JOTED)*, pages 81–97, 2016.

[27] Tashiro T. Speriosu M. Comparison of okapi bm25 and language modeling algorithms for ntcir-6. *Justsystems Corporation*, 14, 09 2006.

[28] Edy Suwondo. *LCC-OPS: Life Cycle Cost Application in Aircraft Operations*. PhD thesis, University of Technology Delft, 2007.

[29] To70. To70 Civil Aviation Safety Review 2017, 2018. URL http://to70.com/to70s-civil-aviation-safety-review-2017/. Date Accessed: 04-04-2018.

[30] Howard R. Turtle and W. Bruce Croft. A comparison of text retrieval models. *The Computer Journal - Special issue on information retrieval*, 35(3):279–290, June 1992. ISSN 0010-4620. doi: 10.1093/comjnl/35.3.279. URL http://dx.doi.org/10.1093/comjnl/35.3.279.

[31] W.J.C. Verhagen and R. Curran. An ontology-based approach for aircraft maintenance task support. *20th ISPE International Conference on Concurrent Engineering, CE 2013 - Proceedings*, pages 494–506, 01 2013.

[32] Rob Wylie, Robert Orchard, Michael Halasz, and François Dubé. Ids: Improving aircraft fleet maintenance. In *AAAI/IAAI*, pages 1078–1085, 1997.

[33] Herimanana Zafiharimalala, David Robin, and André Tricot. Why aircraft maintenance technicians sometimes do not use their maintenance documents: Towards a new qualitative perspective. *The International Journal of Aviation Psychology*, 24(3):190–209, 2014. doi: 10.1080/10508414.2014.918444.

A

Appendix A - Python code

```
#!/usr/bin/env python
import time, re
from myfunctions import *

start_time = time.time()            # Start timer

user_queries = ['wing', 'test']                 # Place where user queries have to be filled in
queries = []                                    # Empty list to place filtered queries
for user_query in user_queries:                 # Filter and append every user query
    queries.append(filter(user_query))


directory = '/Users/guusgeurts/PycharmProjects/Thesis_draft/Macster/Crawler output'      # Path to files
selectTopFiles = 5                                                                      # Top N files for feedback
    calculation
selectTopWords = 10                                                                     # Top M words for
    feedback calculation

(docLength, numberOfDocuments, invindex, totaluniquewords, total_words_per_file) = importFiles(directory,
    debug=True)
avgdl = float(docLength / numberOfDocuments)            # Average document length (avg number of documents
    per file)


# Calculate the BM25 score of all documents
bm25s = calculation(queries, avgdl, numberOfDocuments, invindex, totaluniquewords, total_words_per_file,
    debug=True)

relevantFiles = []
for key, value in sorted(bm25s.iteritems(), key=lambda(k, v): (v, k), reverse=True):
    print("Document %s has a total BM25 score of %s for all queries together." % (key, value))
    relevantFiles.append(key)


# PART FOR FEEDBACK CAN BE FOUND BELOW #

newQueries = queries                                            # Take over original queries

# Append from the top files the words with highest score for TF-IDF and append them to the original query
for key, value in idfPerTotalUniqueWordsFeedback(\
                    numberOfDocumentsPerTotalUniqueWords(\
                        relevantFiles[:selectTopFiles], numberOfDocuments, invindex, totaluniquewords), \
                    numberOfDocuments)[:selectTopWords]:
    newQueries.append(key)

print(newQueries)


# Calculate the BM25 score for all files, but now with new queries (from after feedback).
```

```
bm25s = calculation(newQueries, avgdl, numberOfDocuments, invindex, totaluniquewords, total_words_per_file
    , debug=False)
for key, value in sorted(bm25s.iteritems(), key=lambda(k, v): (v, k), reverse=True):
    print("Document %s has a BM25 Score %s after feedback" % (key, value))
    relevantFiles.append(key)

print("--- %s seconds ---" % (time.time() - start_time))     # End timer and print time to run program

#print(len(totaluniquewords))
#print(totaluniquewords)
```

```
# All modules that need to be imported
import os, math, re
from bs4 import BeautifulSoup
from collections import Counter, defaultdict
from operator import itemgetter


# Function to filter html tags and punctuation.
def filter(string):
    result = string\
        .lower()\
        .replace("(", " ( ")\
        .replace(")", " ) ")\
        .replace(",", " ")\
        .replace(".", " ")\
        .replace(":", " ")\
        .replace("</div>", " </div> ")\
        .replace("</span>", " </span ")\
        .replace("</thead>", " </thead> ")\
        .replace("</table>", " </table> ")\
        .replace("table>", "table")\
        .replace("</td>", " </td> ")\
        .replace("</th>", " </th> ")\
        .replace("</tr>", " </tr> ")\
        .replace("/ctl", " /ctl ")\
        .replace("c/b-", " c/b- ")\
        .replace("<a>", " <a> ")\
        .replace("<b>", " <b> ")\
        .replace("<i>", " <i> ")\
        .replace("p/bsw-", " p/bsw- ")\
        .replace("bsw-", "bsw")\
        .replace("oat=", "oat") \
        .replace('"', ' ')\
        .replace("'\u25b8'", " test1234 ")\
        .replace("fonctional", "functional")\
        .replace("/", " / ")
    result = re.sub(r'[-!;><?.,]$', '', result)      # Remove characters listed in [brackets], only if
        last characters
    return result

# Function to import the HTML files.
def importFiles(directory, debug=False):
    invindex = []                                   # Empty list for inverted index
    totalUniqueWords = defaultdict(lambda: 0)       # Empty dictionary for total number of unique words in all
        files
    totalWordsPerFile = {}                          # Empty dictionary for total number of words per file

    files = os.listdir(directory)                   # List with names of files in directory
    total_num_words_all_files = 0                   # Set total number of words in all files to 0
    for file in files:
        path = os.path.join(directory, file)                    # Path to specific document
        document = open(path, "r")                              # Open specific document
        soup = BeautifulSoup(filter(document.read()), 'lxml')   # Read specific document and filter
        words = soup.text.split()                               # Separate words in document
        uniquewords = Counter(words).keys()                     # "Names of" unique words per file
        values = Counter(words).values()                        # Number of unique words per file
        for uniqueword in uniquewords:
            if uniqueword != totalUniqueWords:                  # If the word is not in dictionary of
                total unique words
```

```
57                  totalUniqueWords[uniqueword] += 1                # Add the word to the dictionary
58
59          for i in xrange(0, len(uniquewords)):                    # For all unique words
60              invindex.append([uniquewords[i], file, values[i]])   # Make an inverted index [word, file where
                    word was found, number of times word is found]
61          if debug:
62              print("Document %s has %s unique words and %s total number of words" % (file, len(uniquewords)
                    , len(words)))
63          totalWordsPerFile[file] = len(words)                     # Dictionary with file name + total number
                of words in this file
64          total_num_words_all_files += len(words)                  # Total number of words in all files
65
66      if debug:                                                    # If debug is "True" print ---
67          print("----------------------")
68      return total_num_words_all_files, len(files), invindex, totalUniqueWords, totalWordsPerFile
69
70  def calculation(queries, avgdl, numberOfDocuments, invindex, totaluniquewords, total_words_per_file, debug
        =True):
71      # Part to calculate document specific values
72      word_idf = {}                                                # Empty dictionary for word inverse document
            frequency
73      bm25s = {}                                                   # Empty dictionary for new BM25 calculation
74      for query in queries:
75
76          results = []                                         # Empty list for match between word in inverted
                index and query
77          other_words = []                                     # Empty list if query is part of a word in
                inverted index
78          for inv in invindex:
79              if query == inv[0]:
80                  results.append(inv)                          # If word in inverted index is equal to query. Add
                        to results.
81              if query in inv[0] and not query == inv[0]:      # If query is part of word (example: arg is
                    inside argument)
82                  other_words.append(inv)                      # Add this word to other_words
83
84          similar_words = []                                               # Empty list for similar words
85          for item in other_words:
86              if item[0] not in similar_words and not item[0] == query:    # If word is not in list and not
                    equal to query
87                  similar_words.append(item[0])                            # Add word to similar_words
88
89          num_doc_contain_query = 0                                            # Set number of documents to
                0
90          for result in sorted(results, key=itemgetter(2), reverse=True):      # For all matches of query
91              if debug:
92                  print(" '%s' found in %s (%s hits!)" % (result[0], result[1], result[2]))   # Print ("
                        query", file, number of hits per file)
93              if result[2] >= 1:
94                  num_doc_contain_query += 1       # Add 1 to number of documents containing the query
95          if num_doc_contain_query > 0:
96              if len(similar_words) >= 1:
97                  if debug:
98                      print("Be careful, these words are almost similar: %s" % similar_words)  # Print when
                            similar words found
99
100             for word in totaluniquewords:                                    # Calculate tf-idf of every query
101                 word_idf[query] = math.log10(
102                     float((numberOfDocuments - num_doc_contain_query + 0.5)) / float(0.5 +
                            num_doc_contain_query))
103             if debug:                                                        # If debug "True" print
104                 print("The Inverse Document Frequency for '%s' is %s" % (query, word_idf[query]))
105         else:
106             word_idf[query] = math.log10(
107                 float((numberOfDocuments - num_doc_contain_query + 0.5)) / float(0.5 +
                        num_doc_contain_query))
108             if debug:
109                 print("%s not found anywhere! The Inverse Document Frequency is: %s" % (query, word_idf[
                        query]))
110
111             # Part to calculate BM25 of query for each document
```

```
112         k1 = 1.5  # 1.2 < k < 2
113         b = 0.5  # 0.5 < b < 0.8
114
115         bm25 = 0
116         for item in sorted(results, key=itemgetter(2), reverse=True):
117             bm25 = word_idf[query] * item[2] * (k1 + 1) / (
118             item[2] + k1 * (1 - b + b * total_words_per_file[item[1]] / avgdl))
119             # print (len(uniquewordsperfile[item[1]]))
120             if debug:
121                 print("For the term: '%s' the BM25 Score of %s is %s" % (query, item[1], bm25))
122             if item[1] in bm25s:
123                 bm25s[item[1]] = bm25s[item[1]] + bm25
124             else:
125                 bm25s[item[1]] = bm25
126
127         if debug:
128             print("----------------------")
129     return bm25s
130
131
132 def numberOfDocumentsPerTotalUniqueWords(files, numberOfDocuments, invindex, totaluniquewords):
133     allWords = []
134     result = {}
135     for inv in sorted(invindex):
136         if inv[1] in files:
137             allWords.append(inv[0])
138
139     for word in Counter(allWords).keys():
140         result[word] = 0
141         for inv in invindex:
142             if inv[0] == word:
143                 result[word] = result[word] + 1
144     return result
145
146 def idfPerTotalUniqueWordsFeedback(numberOfDocumentsPerTotalUniqueWords, numberOfDocuments):
147     result = {}
148     for uniqueWord in numberOfDocumentsPerTotalUniqueWords:
149         result[uniqueWord] = math.log10(
150                     float((numberOfDocuments - numberOfDocumentsPerTotalUniqueWords[uniqueWord] + 0.5)) /
                                float(0.5 + numberOfDocumentsPerTotalUniqueWords[uniqueWord]))
151     return sorted(result.iteritems(), key=lambda(k, v): (v, k), reverse=True)
```

B

Appendix B - Critical Values of the Wilcoxon Signed-Rank test

Critical Values of the Wilcoxon Signed Ranks Test

n	Two-Tailed Test		One-Tailed Test	
	α = .05	α = .01	α = .05	α = .01
5	--	--	0	--
6	0	--	2	--
7	2	--	3	0
8	3	0	5	1
9	5	1	8	3
10	8	3	10	5
11	10	5	13	7
12	13	7	17	9
13	17	9	21	12
14	21	12	25	15
15	25	15	30	19
16	29	19	35	23
17	34	23	41	27
18	40	27	47	32
19	46	32	53	37
20	52	37	60	43
21	58	42	67	49
22	65	48	75	55
23	73	54	83	62
24	81	61	91	69
25	89	68	100	76
26	98	75	110	84
27	107	83	119	92
28	116	91	130	101
29	126	100	140	110
30	137	109	151	120