Tracing requirement objects as an information retrieval task

Richárd Lengyel

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 30.7.2020

Thesis supervisor:

Prof. Antti Oulasvirta

Thesis advisors:

D.Sc. (Tech.) Luis Leiva

M.Sc. Jaakko Hujanen

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Richárd Lengyel

Title: Tracing requirement objects as an information retrieval task

Date: 30.7.2020 Language: English Number of pages: 8+66

Department of Electrical Engineering and Automation

Professorship: Autonomous Systems

Supervisor: Prof. Antti Oulasvirta

Advisors: D.Sc. (Tech.) Luis Leiva, M.Sc. Jaakko Hujanen

In large requirement databases, tracing of different objects to each other, e.g. higher-level requirements to lower-level requirements, or requirements to their verification methods, can be a tedious job. With numerous objects in the database the selection of the corresponding object from lists can take a long time. In this thesis standard information retrieval (IR) methods, in particular multiple variants of vector space modelling, are applied in order to provide a shortlist of a few objects, which are predicted to be relevant, this way speeding up the selection process.
The aim of the thesis is to demonstrate the usage of such IR system on a real-life example requirement data set, providing an end-to-end solution from processing the relevant data to showing the shortlist on a GUI view. The separation of the data to train and validation subsets and the setup of a relevant evaluation metric is also essential in order to benchmark future developments.

Keywords: Information retrieval, Vector space model, Requirements tracing, Automated requirement tracing

Acknowledgements

I would like to say thank you for the continuous assistance during the thesis process to Jaakko, Luis and Antti.

Also, to my wife, Markéta for all her patience and support.

Otaniemi, 30.7.2020

Richárd Lengyel

Contents

Symbols and abbreviations

Abbreviations

IDF	Inverse Document Frequency
IR	Information Retrieval
GUI	Graphical User Interface
RE	Requirements Engineering
TF	Term Frequency

1 Introduction

In the following section the overall environment is described, in which this thesis and the related software development was performed. The reader can get a basic understanding of the project's background and the business goal behind the thesis topic.

1.1 Background: Autoport project

The Autoport project is a consortium of multiple Finnish industrial companies, such as Exertus, Kalmar and Huld (previously Space Systems Finland, SSF) and research institutes, as Tampere University and VTT, aiming to bring innovative solutions to the marine industry. [1] The ecosystem's main goal is to develop safe, reliable and cost effective solutions towards an autonomous port.

The project has multiple sub-projects, some of them are connected to the core innovation process of autonomous harbours, other sub-projects are working on loosely-coupled accompanying activities. One of these is the Autoport Tool, described in detail at 1.1.1. Besides the industrial acitivities, the project also served as inspiration for research papers, such as [2] and [3].

1.1.1 The Autoport Tool

The Autoport Tool is a sub-project of the Autoport project. It is a data-based safety requirement management software, developed by Huld within the Autoport ecosystem. The Tool is aimed to be generic enough to be used in different corporations' safety management life-cycle, as a safety or general requirements management tool. Therefore it shall be able to keep track of different objects, their changes and relations to eachoter.

In order to achieve this goal, the Tool enables its users to create, update, delete different object and link types. In the safety management domain, these object types can be, among others, hazards, their mitigation, requirements or safety functions. In general software development projects the object types could be customer or software requirements and their verification plans (test specifications), in the development phase of this thesis, a similar data set was used for hyperparameter tuning and performance evaluation. Thus, in the following, the reader should recall different requirements and test specifications when object types are mentioned. More detailed at 1.2.4.

Link types are described between the previously described object types. If one would like to create a link type between hazards and their mitigation process, 'mitigates' could be a sufficient naming. Similarly for software requirements and their test specifications 'tests' could be used.

Once these object types and link types are created, the next step is to populate the Tool's database with the actual data, so different requirements and tests are added at this stage.

1.2 Problem Definition

This section is meant to describe the specific role of the thesis within the project. It states the business goal behind the research and also mentions on a high level, the applied scientific fields and technologies.

1.2.1 Difficulties when number of requirements grow

The previously described task, tracing of requirement objects to each other, can be tedious. Practically on a GUI, when one is setting up a software requirement and selecting the other relevant requirements or test to be linked to this requirement, it can take time to select these matching object for example from a drop-down list. When the number of objects in a given object type reaches just around hundred - which is completely realistic for even medium sized projects, similar to the example data set - this increase in selection time is already observable. On top of that, with a high number of options the selection is also prone to errors.

1.2.2 Automated tracing of requirements

The focus of this thesis is to offer a shortlist of relevant objects, related to a query object - the object to be traced from - speeding up the tracing process this way. The problem of getting a shortlist of n related objects can be formulated as an information retrieval (IR) task. Each object (a requirement or a test specification) has a text field, which is embedded into a high-dimensional vector space and then the similarity of these vectors are measured with a certain similarity metric. The top n objects, which belong to the most similar vectors compared to the query vector, will be returned as a shortlist.

1.2.3 Information retrieval and computational GUI design optimization

Although most of the tools used to prepare the shortlist belong to IR, the problem can also be interpreted as a computational optimization of the GUI designs, as the goal is to have an optimal menu layout. Some techniques, as Fitt's law, are therefore also used during the desing process.

1.2.4 Exomars project as source of example data set

In the initial IR system, Huld's example data set was used. The data set is related to the company's contribution to the ExoMars project[4] and consists of a few hundred objects.

The ExoMars project is a cooperation between ESA and Roscosmos, and the aim of the project is to search for the signs of life on Mars. Huld's work was related to recovery software of the rover, which is resposible for the initialization and back-up of the system. The example data set used in the thesis is derived from the Software Requirements Specification (SRS) and from the test specifications of this software project.

The derived objects were divided into the previously mentioned three object types: customer requirements, software requirements and test specifications. The customer requirements were provided by Airbus, the software requirements are derived from these, further detailing some of them, while test specifications are meant to validate the customer and software requirements. The latter two belong to Huld's contribution.

2 Related work

The thesis largely builds on three separate main fields of computer science:

1. Requirements engineering (RE)
2. Information retrieval (IR)
3. Computational UI design

Requirement engineering is important in order to understand the context of the application and the business value it generates. Information retrieval does the actual job and it is responsible for the core functionality of the application, while computational UI design gives another point of view on the optimization problem under discussion. In the following, a short introduction to the relevant parts of all main fields will be presented.

2.1 Requirements specification and management

This section serves as an introduction into the different tasks and related processes of requirements engineering, as well as a clarification on its importance.

2.1.1 Mission/safety critical projects

Following Fowler's definition [5] a safety-critical system is a system that creates a physical hazard to human life. If a failure takes place in such a system, it can cause physical harm to user or other humans interacting or being in the vicinity of the system.

Wheres, a mission critical system is such, where the failure of the system can lead to the failure of the operation or company.

In the current thesis, a safety/mission critical project is a project, which develops safety/mission critical system.

These projects have several attributes, which are unavoidable in order to ensure the success of the projects. These attributes are not necessarily unique to the previously mentioned type of projects - other projects might use the same processes - but are essential in mission critical projects. The most important such attributes are detailed in the following.

2.1.2 Exhaustive testing

An important attribute of safety critical projects is the exhausting testing of all requirements. Naturally, on the lowest level, unit testing is also executed, but after that all software functionality has to have a verification plan. Generally speaking, that could be a calculation, test, code review, analysis of design. In software projects the verification plan is prominently testing. Which means, that all software requirements have to be linked with test plans and their results. This is one of the main drivers for tracing the requirements, detailed at 2.1.3.

2.1.3 Tracing requirements

In requirements management, it is of utmost importance to be able to trace between different levels of requirements, e.g. from which customer requirement is a given software requirement derived, or between requirements and their verification plans, such as tests, calculations, etc. This way, engineers can always see the source object of any object in the database. Furthermore when a change is applied at any level of the overall system, it is easy to trace which other objects were affected.

The figure below is meant to aid with illustrating the tracing of requirements and the corresponding change management. The three object types' instances are shown as circles, with a number referring to a particular object. As it is visible on the figure, objects of different types are traced, linked to each other. The typical case is that from a customer requirement a software requirement is derived, which later on associated with a test specification, but this is not the only possibility, as the linking between the objects is a many-to-many type relation. The figure also shows, that if a customer requirement is changes, in the current example the customer requirement number one, then it is flagged and the flagging is propagated through the tracing, flagging each associated object too.

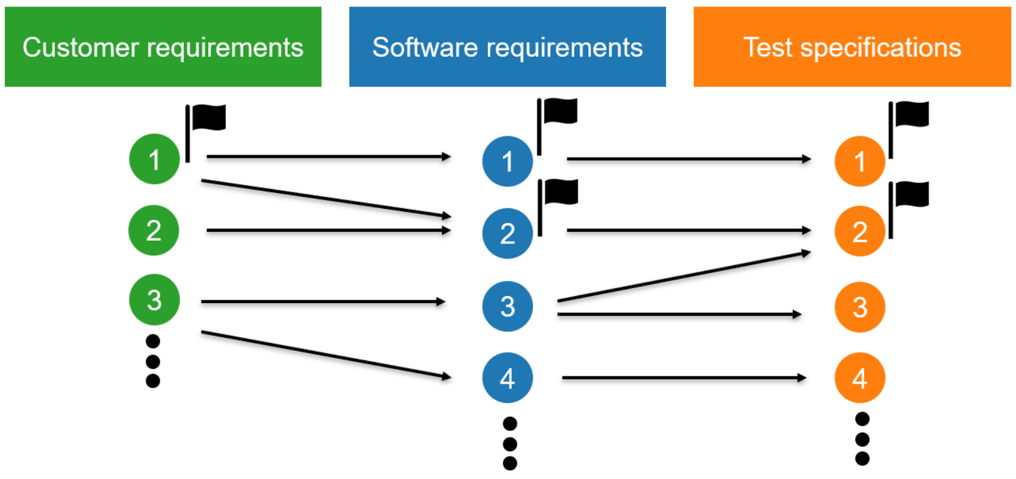


Figure 1: Requirements tracing and change management

The Center for Disease Control and Prevention in a 2006 paper [6] explains the essence of a requirement tracing system and also gives guidelines to the proper usage of requirement tracing. The paper states that tracing requirements can ensure that all steps of the development process are executed according to the original needs and requirements of the customer. Furthermore, they are compliant with all other kinds of functional or quality requirements of the design.

Regarding the guidelines, the most important attributes are, that all requirements have a unique identification in order to warrant traceability. The bidirectional nature of linkage between objects is also key, as it allows tracing in both directions. The verification of all requirements was mentioned previously, and the history of requirement changes will be detailed in 2.1.4. Both of these are essential parts of any requirement tracing systems.

2.1.4 Change management

In order to make sure that, things are under control during the whole life cycle of a safety/mission critical project, the set up of a change management system is unavoidable. Software configuration management, one of the key practices of these systems, helps the developers and the management to track and coordinate changes of objects, or as it is often called in requirements engineering, artifacts. It enables the users of the system to keep track of who made changes, on which artifact and when. Furthermore, as tracing artifacts is one of the core functionalities of change management systems, it also shows what other artifacts were affected by the changes.

Mohan et al. [7] suggest that the close integration of the two elemental methods are improving the overall performance of a change managements system.

2.1.5 Verification and validation

Verification, as mentioned previously, ensures that the actual product complies with the previously specified requirements or with other regulations such as standards. Validation on the other hand makes sure the product meets the user's or customer's needs. It actually aims to clarify that the requirements were set up correctly.[8]

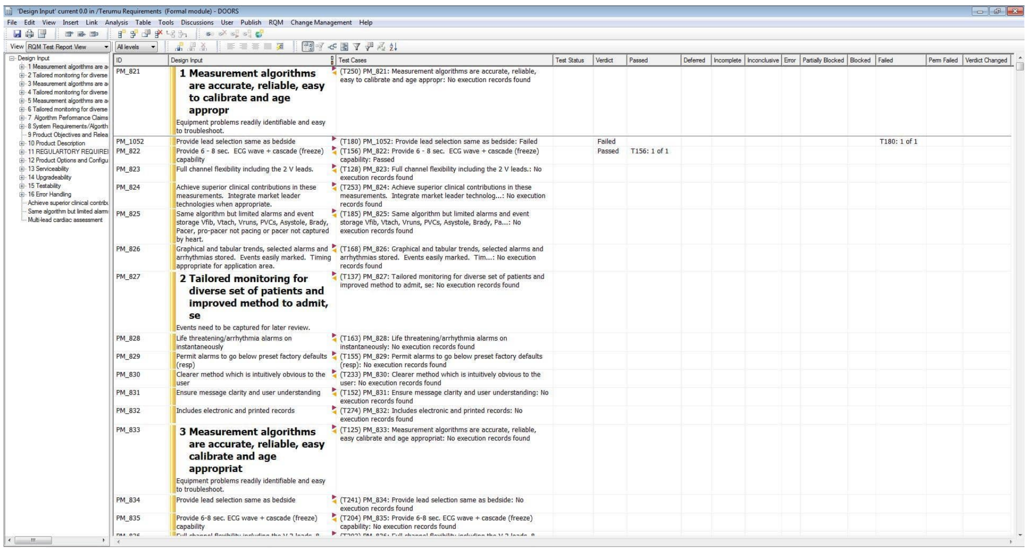
Another expressive way to formulate the difference between verification and validation was given by [9]: verification guarantees that the equations are solved correctly, while validation makes certain that the correct equations are solved.

Naturally in mission/safety critical projects both the verification and validation needs to be handled exhaustively for every requirement.

2.1.6 Requirements management tools

In the following, a short summary is about the available requirements management software tools is shown. The selection is not aimed to be complete, instead the idea is to present three widely-used and significantly different tools shortly, to show how differently the tasks of RE was touched before. All of the listed tools were used by Huld, therefore the following summary is based on a discussion with safety experts at the company.

IBM Engineering Requirements Management DOORS [10] IBM's DOORS is one of the most well-known, traditional requirements management tools available on the market. Its main user interfaces are tabular, showing the the stored data and enabling adding new records in an excel-like fashion. It's function list is almost endless, naturally it supports change management through showing the impact of changes, e.g. by marking the affected rows. It clearly puts the focus of the requirements management part of the life-cycle and stresses the linking to tests less.



'Design Input' current 0.0 in /Terumu Requirements (Formal module) - DOORS

File Edit View Insert Link Analysis Table Tools Discussions User Publish RQM Change Management Help

View: RQM Test Report View — All levels

- Design Input
  - 1 Measurement algorithms are a
  - 2 Tailored monitoring for diverse
  - 3 Measurement algorithms are a
  - 4 Tailored monitoring for diverse
  - 5 Measurement algorithms are a
  - 6 Tailored monitoring for diverse
  - 7 Algorithm Performance Claims
  - 8 System Requirements/Algorith
  - 9 Product Objectives and Relea
  - 10 Product Description
  - 11 REGULARTORY REQUIREI
  - 12 Product Options and Configu
  - 13 Serviceability
  - 14 Upgradeability
  - 15 Testability
  - 16 Error Handling
  - Achieve superior clinical contribu
  - Same algorithm but limited alarm
  - Multi-lead cardiac assessment

ID	Design Input	Test Cases	Test Status	Verdict	Passed	Deferred	Incomplete	Inconclusive	Error	Partially Blocked	Blocked	Failed	Perm Failed	Verdict Changed
PM_821	**1 Measurement algorithms are accurate, reliable, easy to calibrate and age appropr** Equipment problems readily identifiable and easy to troubleshoot.	(T250) PM_821: Measurement algorithms are accurate, reliable, easy to calibrate and age appropr: No execution records found												
PM_1052	Provide lead selection same as bedside	(T180) PM_1052: Provide lead selection same as bedside: Failed		Failed								T180: 1 of 1		
PM_822	Provide 6 - 8 sec. ECG wave + cascade (freeze) capability	(T156) PM_822: Provide 6 - 8 sec. ECG wave + cascade (freeze) capability: Passed		Passed	T156: 1 of 1									
PM_823	Full channel flexibility including the 2 V leads.	(T128) PM_823: Full channel flexibility including the 2 V leads.: No execution records found												
PM_824	Achieve superior clinical contributions in these measurements. Integrate market leader technologies when appropriate.	(T253) PM_824: Achieve superior clinical contributions in these measurements. Integrate market leader technolog...: No execution records found												
PM_825	Same algorithm but limited alarms and event storage Vfib, Vtach, Vruns, PVCs, Asystole, Brady, Pacer, pro-pacer not pacing or pacer not captured by heart.	(T185) PM_825: Same algorithm but limited alarms and event storage Vfib, Vtach, Vruns, PVCs, Asystole, Brady, Pa...: No execution records found												
PM_826	Graphical and tabular trends, selected alarms and arrhythmias stored. Events easily marked. Timing appropriate for application area.	(T168) PM_826: Graphical and tabular trends, selected alarms and arrhythmias stored. Events easily marked. Tim...: No execution records found												
PM_827	**2 Tailored monitoring for diverse set of patients and improved method to admit, se** Events need to be captured for later review.	(T137) PM_827: Tailored monitoring for diverse set of patients and improved method to admit, se: No execution records found												
PM_828	Life threatening/arrhythmia alarms on instantaneously	(T163) PM_828: Life threatening/arrhythmia alarms on instantaneously: No execution records found												
PM_829	Permit alarms to go below preset factory defaults (resp)	(T155) PM_829: Permit alarms to go below preset factory defaults (resp): No execution records found												
PM_830	Clearer method which is intuitively obvious to the user	(T233) PM_830: Clearer method which is intuitively obvious to the user: No execution records found												
PM_831	Ensure message clarity and user understanding	(T152) PM_831: Ensure message clarity and user understanding: No execution records found												
PM_832	Includes electronic and printed records	(T274) PM_832: Includes electronic and printed records: No execution records found												
PM_833	**3 Measurement algorithms are accurate, reliable, easy calibrate and age appropriat** Equipment problems readily identifiable and easy to troubleshoot.	(T125) PM_833: Measurement algorithms are accurate, reliable, easy calibrate and age appropriat: No execution records found												
PM_834	Provide lead selection same as bedside	(T241) PM_834: Provide lead selection same as bedside: No execution records found												
PM_835	Provide 6-8 sec. ECG wave + cascade (freeze) capability	(T204) PM_835: Provide 6-8 sec. ECG wave + cascade (freeze) capability: No execution records found												

Figure 2: IBM Doors example tabular UI view[11]

Visure Requirements Management ALM platform [**12**] Visure is a widely used RE tool, keeping the big picture of requirements management under control, not focusing merely on the storage and linkage of requirements, but its aim is to get the whole life-cycle managed within a single program. A good example for this is that test specifications and their acceptance results are easily available for each requirement. Besides the traditional tabular view, it applies some more visual elements, for example in order to define workflows, for example about the acceptance of changes related to RE within the company, one can use a block diagram based editor, illustrated by the fiugre below.

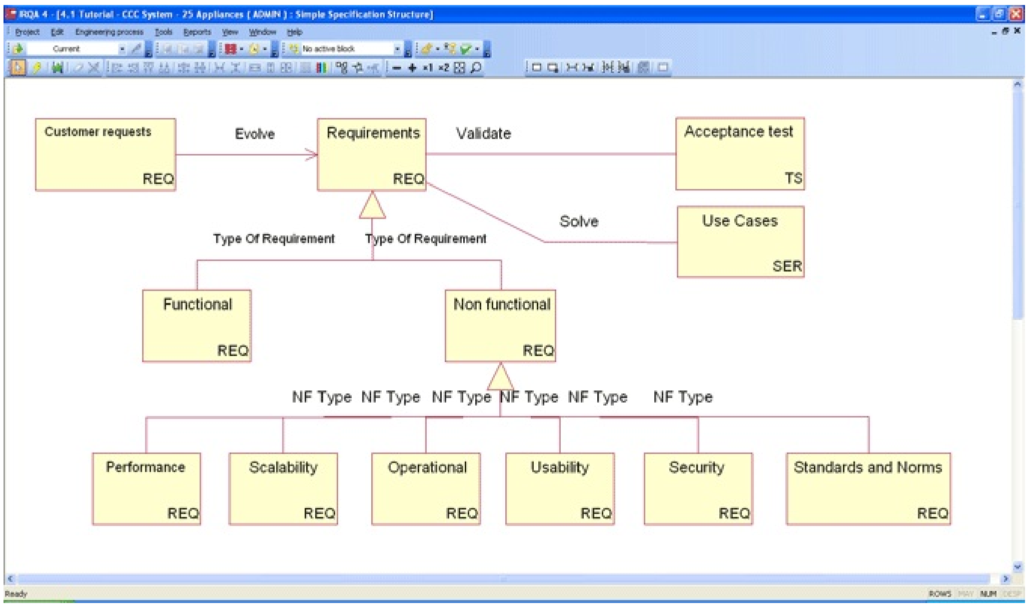
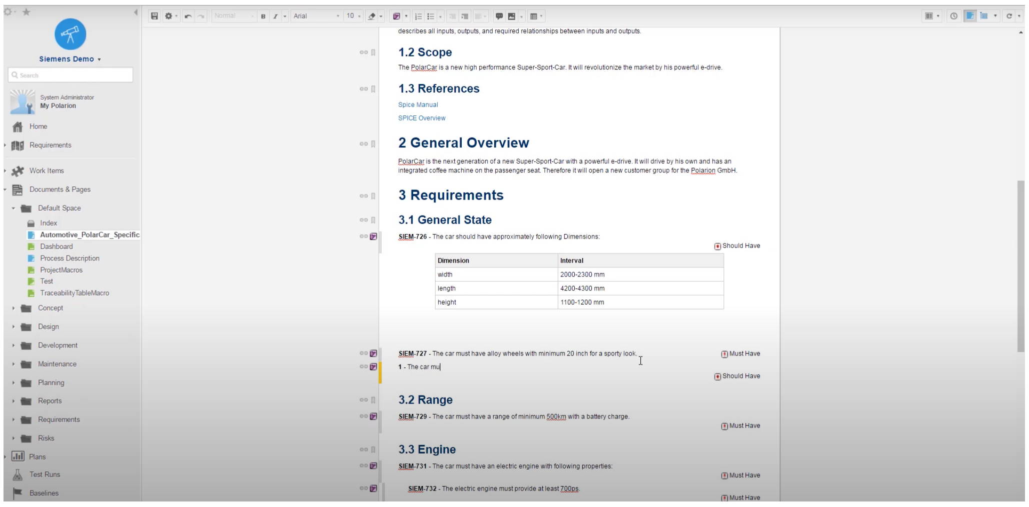


Figure 3: Visure Block Diagram UI view[13]

Polarion Requirements [14] While most RE tools will first populate a database with artifacts through some structured forms, Polarion is using a significantly different approach. Their feature, called LiveDoc, looks like a document editor, where one can keep track of the requirements in a free text form, and then this view is converted into artifacts by specifying them in the text by clicking.



describes all inputs, outputs, and required relationships between inputs and outputs.

1.2 Scope

The PolarCar is a new high performance Super-Sport-Car. It will revolutionize the market by his powerful e-drive.

1.3 References

Spice Manual

SPICE Overview

2 General Overview

PolarCar is the next generation of a new Super-Sport-Car with a powerful e-drive. It will drive by his own and has an integrated coffee machine on the passenger seat. Therefore it will open a new customer group for the Polarion GmbH.

3 Requirements

3.1 General State

SIEM-726 - The car should have approximately following Dimensions: Should Have

Dimension	Interval
width	2000-2300 mm
length	4200-4300 mm
height	1100-1200 mm

SIEM-727 - The car must have alloy wheels with minimum 20 inch for a sporty look. Must Have

1 - The car mu Should Have

3.2 Range

SIEM-729 - The car must have a range of minimum 500km with a battery charge. Must Have

3.3 Engine

SIEM-731 - The car must have an electric engine with following properties: Must Have

SIEM-732 - The electric engine must provide at least 700ps. Must Have

Figure 4: Polarion LiveDoc UI view[15]

2.2 Information Retrieval

In [16] Manning, P. Raghavan, and H. Schütze defines information retrieval as finding unstructured material (usually text) from within large repositories of documents, called corpus, that are relevant to an information need expressed by a query.

Further explained, in order to have an IR system, one needs a user, who has some kind of information need - a question to be answered - and one also needs a UI to formulate this question as a query. Then 'at the back' of the IR system, the query is processed and, based on some kind of selection method, the most relevant documents are tried to be filtered and retrieved from the set of all documents in the system.

The section related to IR is mostly based on the work of Manning et al. [16].

2.2.1 Basic terminology

Term In information retrieval the elementary building stones are terms, which are words in the everyday sense. Usually, the words are first pre-processed due to simplification issues.

Document After the terms, the next unit is the document, which is built up from a number of terms. Generally, a document can be e.g. a book or a website. In the current project documents are requirements and test specifications, or in other sections they are also referred to as objects.

Corpus The set of all documents in the repository is called corpus.

Vocabulary The set of all unique terms in the corpus - therefore in all included documents - is the vocabulary or dictionary (of the corpus).

Query The query, as it was mentioned before, is the textual expression of an information need.

Revisiting the definition in the previous section, the goal of information retrieval is to find documents in the corpus, similar to an arbitrary query. The similarity is usually based on the textual content of the documents, so on the terms involved in it.

2.3 Inverted indexing

In order to efficiently be able to search and compare the documents of the corpus, a special representation is used, the so-called inverted indexing. A forward-index would map to each document a list of their involved terms. The inverted index works the other way around, and maps a list to each term of the dictionary, which contains all the documents in which the term occurs.

Each item in the lists are called a posting, and the lists are called therefore a postings list according to the convention.

A statistical metric, later on used by the tf-idf model, is derived directly from the postings. The length of each postings list is the corresponding term's document frequency. Another important metric might be stored in the postings list: the term frequency. If it is stored, then instead of simply storing the id's of the documents in which the term occurs, it is also stored how many times the term occured in the document.

2.3.1 Determining the vocabulary

In order to determine the vocabulary of a corpus, several pre-processing steps are necessary, as the original raw documents are hard to handle and may produce misleading results.

Tokenization The first step is the tokenization. It takes care of slicing up the document into tokens, and oftentimes getting rid of unnecessary characters, such as the punctuation between words and sentences.

Normalization As the documents are sliced into tokens, the next step is token normalization. This step makes an attempt to group tokens if they are not the same character by character, but it would be sensible to treat them as one. In other words normalization takes care of the equivalence-classing of the terms. There can be many occasions when this is advised, for instance words written with upper or lower case characters often should be treated as the same term.

Stop words When building the dictionary and the inverted index, optionally some words are labeled as stop words and they are not added to the vocabulary. Typically those words are eligible for being a stop word, which are too common and too general to have any descriptive power about the documents. For instance "the", "and", "of", etc.

Stemming Stemming can be seen as another form of normalization. It is used, when words are differing due to grammatical reasons - such as e.g. test, test, testing, tested - but have basically a common meaning. In order to treat them as one, stemming cuts the end of the words with the hope of achieving equality.

2.3.2 Zone indexes

Generally, the document is not just a sequence of words, where all parts have similar importance and functionality, rather it can have a main text part, a title, description or other metadata. These different functional parts of the document are called zones. When a query is filled into the system, the queried terms might be looked up and found in different zones of each document.

Imagine, that a term occured in two documents, but in different zones of them. In order to decide which document should be assumed to be more relevant, an importance weight can be assigned to the zones of the documents. The weights should sum up to 1, thus a similar weighting is possible: title 0.4, description 0.3, text 0.3.

In order to determine the optimal weighting, regression models can be applied.

2.3.3 Tf-idf

The term frequency - inverse document frequency, or tf-idf is a key metric used for ranking documents based on their similarity to a query.

$$tfidf = tf \cdot idf \tag{1}$$

It is the product of the term frequency (tf) - how many times the term occurs in the document - and the inverse document frequency (idf) - the inverse of the number which shows in how many documents the term occurs. The higher the tf value is, the more occurences of the terms can be observed in the document, therefore it is clever to assume, that the term is descriptive of the document. On the other hand the term could also be general word, that is why the inverse idf is also important. If the idf score is high, the term occurs only in relatively few documents, this way ensuring that it is not a general word, further strengthening the assumption that it is actually descriptive to the document.

Thanks to the balancing nature of tf and idf in the product, the tf-idf is a very useful metric to determine the descriptive power of term related to a document.

2.4 Boolean retrieval

In case of a traditional IR system, when the queries would be some few number of search terms, the easiest approach to be implemented is the Boolean retrieval.

In the current project, as the queries are also actual documents from the data set - the task is to trace requirements and tests to each other - it is less relevant, as it is not very likely that two documents would be that similar that it would be sensible to run a Boolean retrieval.

Nevertheless, the core idea is explained in this section: a Boolean retrieval model produces queries out of terms by connecting them by the three elementary Boolean operators (and, or, not). Then the retrieval is carried out according to the Boolean rules, e.g. if all terms are connected by the *and* operator then all terms have to be presented in the retrieved documents, while if terms are connected by the *or* operator, then all documents are to be retrieved, which contian at least one of the terms.

2.5 Vector space model

The vector space model is a bag-of-words representation of documents, which means that documents are represented only by the contained terms and not by the position of the terms (relative to each other).

In the vector space model each document is represented by a vector with the dimensionality equal to the length of the vocabulary of the corpus and each dimension takes a metric related to the corresponding term as value. In other words, the documents are embedded into a high dimensional (equal to the length of the vocabulary) vector space.

The actual retrieval is then carried out in the following way: the query is also converted into the high dimensional vector space representation and it distance (or similarity) is calculated compared to all other vector representations based on a selected distance (or similarity) measure. Once all distance (or similarity) values are calculated, the documents with the lowest distance (or highest similarity) are retrieved.

2.5.1 Binary representation

The easiest such representation would be a binary vector, with 1-s for the dimensions, where the term is included in the document, and 0-s for those, where it is not.

2.5.2 Counts vector representation

A more sophisticated approach is the counts vector representation, where the dimensions are not taking binary values anymore, but they show how many times the words appear in the document.

2.5.3 Tf-idf representation

Finally, the tf-idf representation puts the tf-idf score of each term into the vectors of each document. This model has a better ability to describe the relevance of a term compared to a document, thanks to the attributes of the tf-idf statistics.

2.6 Bm25 function

Another bag-of-words retrieval model is the BM25, which gives the following score to document D for query Q, containing the terms $q_1, ..., q_n$[17]:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \frac{f(q_i,D) \cdot (k_1 + 1)}{f(q_i,D) + k_1 \cdot (1 - b + b \cdot \frac{D}{avgdl})} \qquad (2)$$

where $f(q_i, D)$ is the term frequency in the document D, $|D|$ is the length of the document D in words, and $avdl$ is the average document length in the corpus. k_1 and b are free parameters.

2.7 Word embeddings

Nowadays, a new, popular direction replacing the bag-of-words is the so called word embeddings. This approach takes each term of a document and embeds it to a high dimensional vector space in a way that semantically similar words are supposed to have vector representations with a smaller distance between each other. This is an important improvement compared to the bag-of-word methods, where only the syntactical similarity played a role. In order to get similar vector representation for words with a similar semantic meaning, neural networks are used. One of the most well-known such models is the Word2vec [18], which takes a large text corpus as input and produces the high dimensional vector space of words appearing in the corpus with the previously mention distance attributes.

When documents need to be represented, one way to go is to first embed each term in the document, and then represent the document by some kind of (weighted) average of the terms in the document. The simplest version just takes the arithmetic average of the words, but this approach does not take into account that different words can have different values in the text. Therefore, a more advanced way is to average the words weighted by their tf-idf score, which captures the value of the words, as it was explained before.

Examples of state-of-the-art document embedding systems are Facebook's InferSent [19] and Julian Brendl's Word2doc [20].

2.8 Distance measures

In any information retrieval task, such as the current requirement tracing, the comparison of objects is essential. In order to define clearly the (dis)similarity of objects, one needs to use the so-called distance functions.

In order to find the most similar objects to a selected query object, one has to define and apply a distance function between pairs of objects yielding a real number - currently represented as vectors.

In more formal, mathematical way, if X is the space of data objects, then the distance function d is as follows:

$$d : X \times X \to \mathbb{R} \tag{3}$$

The larger the value d is for a given pair of objects, the more distant they are considered to be.

2.8.1 Distance metrics

A distance measure is called a distance metrics if it fulfills the following conditions[21] (assuming that $x, y, z \in X$):

1. $d(x, y) \geq 0$: separation axiom, no negative distances are possible
2. $d(x, y) = 0$ if and only if $x = y$: coincidence axiom, distances are positive, except for an object to itself
3. $d(x, y) = d(y, x)$: distances are symmetric
4. $d(x, y) \leq d(x, z) + d(z, y)$: triangle inequality, the direct distance between two objects cannot be greater than the sum of distances with an intermediate point

Some algorithms require the selected distance measure to be a metric, therefore, when it is possible it is recommended to use a metric as distance measure.

2.8.2 Distance and similarity functions

Distance measures, as defined previously, are mapping a pair of data object to a real number, and the higher value they provide the more distant the examined objects are considered to be. Thus, if the function yields zero, the two objects are considered to be identical.

Based on a similar logic, another type of measures can be created, the so-called similarity measures. They practically express the same relation between object as the distance measures do, making it on an inverted way. It is often a convention that a similarity function maps to $[0, 1]$, where one means perfect identity and zero stands for complete difference.

Distance and similarity function are generally easily transformable to one another, especially if the distance function is also yielding results on a bounded $[0, D]$ interval (smaller means more similar). Then, the transformation is simply as follows:

$$s_d = 1 - \frac{1}{D} \tag{4}$$

If the distance function is not bounded, the transformation is still feasible, but one has to choose a function which decays to zero as the distance goes to infinity. For instance, the following is a suitable solution:

$$s_d = e^{\frac{-d}{a}} \tag{5}$$

where a adjusts the speed of the decay.

2.8.3 Examples

In the following, some actual example distance functions are introduced along with a short explanation of their usage.

L_p (Minkowski) distances If the objects x and y are points in the Eucledian space $\mathbb{R}^m$, which also means that they are interpreted as vectors of this space, the L_p distances look the following in the general case:

$$L_p(x, y) = \left(\sum_{j=1}^{m} |x_j - y_j|^p \right)^{\frac{1}{p}} \tag{6}$$

In the following, three notable special cases of the L_p distance is introduced.

- L_1 - Manhattan distance
  If the parameter $p = 1$, the L_p distance simply yields the sum of the absolute value of the differences along each dimension. It is called Manhattan distance, because it is the distance one would have to travel along the house-blocks, typical for Manhattan.
- L_2 - Eucledian distance
  If the parameter $p = 2$, the L_p distance is the well-known Eucledian distance.
- L_∞ - "$L - infinity$" distance
  If the parameter $p = \infty$, the L_p distance is expressed in the following way: $L_\infty(x, y) = \max\limits_{j=1..m} |x_j - y_j|$, which means that the distance is simply equal to the absolute value of the difference along the dimension with the highest difference.

Dot-product similarity If the examined objects are still representable as (equal-length) vectors, then their dot product can be used as a similarity function.

$$d_{dot}(x, y) = \underline{x} \cdot \underline{y} \tag{7}$$

Cosine similarity Cosine similarity is derived from the dot-product similarity, by diving it by the product of the magnitudes of the vectors, this way scaling it to the $[0, 1]$ interval:

$$d_{dot}(x, y) = \frac{\underline{x} \cdot \underline{y}}{|\underline{x}| \cdot |\underline{y}|} \tag{8}$$

As a result the cosine similarity is insensitive to the differences in the magnitude of the vectors and only the difference of their orientation matters.

Others Numerous other distance (and similarity) functions could be defined, but in most cases, when the data is represented as equal-length vectors, the above examples are sufficient. For different-length vectors or completely different input object types, the introduction of new distance functions might be needed, but as the thesis deals with equal-length vector representation of objects, they will not be discussed here.

2.9 Computational interaction design

Computational interaction as defined in [22], is a method, that supports the understanding and improvement of human-computer interaction - using algorithms and mathematical models. It builds on the approaches like updating the model based on observed data, adapting the design based on an algorithm, automating the design process or simulation of user behaviour.

Computational interaction offers various benefits over traditional human-computer interaction design, such as increasing efficiency and satisfaction of the usage of interfaces. It reaches these improvements through formulating abstract goals into mathematical optimization problems supported by real user data. The mathematical formulation can also help arguing for a specific design choice by offering proofs for its validity. When it is used correctly, the user data can ensure that the user stands in the center of focus.

Furthermore, it can reduce design time by automatizing some part of the design, which also has an additional benefit by supporting the designer to focus on the more creative and challenging parts of it.

2.9.1 Fitts's law for menu optimization

In the paper of Oulasvirta et al. [23], and also previously in Ahlström's publication [24], Fitts's law is used to calculate the time needed to reach a specific menu element. The same approach can be used here to determinate the optimal number of returned shortlist elements.

After calculating the shortlist, the proportion of relevant retrievals for the n^{th} returned list element in the previous section, we can use Fitts's law as an optimization criteria to minimize the time required to reach the desired element.

The Shannon formulation, one of Fitts's law's variants, looks the following way [25]:

$$T = a + b \cdot log_2 \left(\frac{D}{W} + 1 \right) \tag{9}$$

where:

- T is the average time needed to complete a movement to an object
- W is the width of the target along the axis of motion
- D is the distance from the starting point to the middle of the obejct
- Furthermore, a and b are parameters determined empirically by linear regression. These values depend on the choice of input device.

When Fitts's law is used alone, the value of the constants does not change where the optimum would be found. However, if it is used in relation to other equations, containing other constants, the optimum can actually depend on their value.

Therefore, in order to get the empirical parameters to the above equation, either the experiment shall be conducted to measure them for a specific setting, or some previous results can be used. As the main focus of the thesis is not related to these kind of empirical experiments, this section serves more as an outlook for future possibilities. In [26] Fitts's law was used to compare the performance of different pointing devices. In [27], the previous results were revised, as the above mentioned Shannon formulation got more accepted compared to previously used variants.

In this thesis the usage of Fitts' law is only mentioned as a possible alternative, therefore the exact values of the constans are not relevant.

3 System architecture

3.1 Overview

The overall system architecture is divided into two separately functioning subsystems: the requirements management application itself and the IR prototype tool. The first one is developed in C++, while the other is developed in Python in order to quickly show results on a proof concept level. Both subsystems are introduced here, and the options for future integration or interfacing is also discussed. As the topic of the thesis is the Python prototype, that section is more detailed, the description of the structure of Autoport tool is simplified and only serves introductory purposes.

3.2 Autoport tool application structure

The Autoport tool, as described before is a requirements management tool developed within the Autoport project. It is decided to be a desktop application over a web-based tool, for performance and privacy issues.

3.2.1 Technology selection

Database and Back-end The tool's back-end technology selection consist of MsSql for the database and C++ for the related operations.

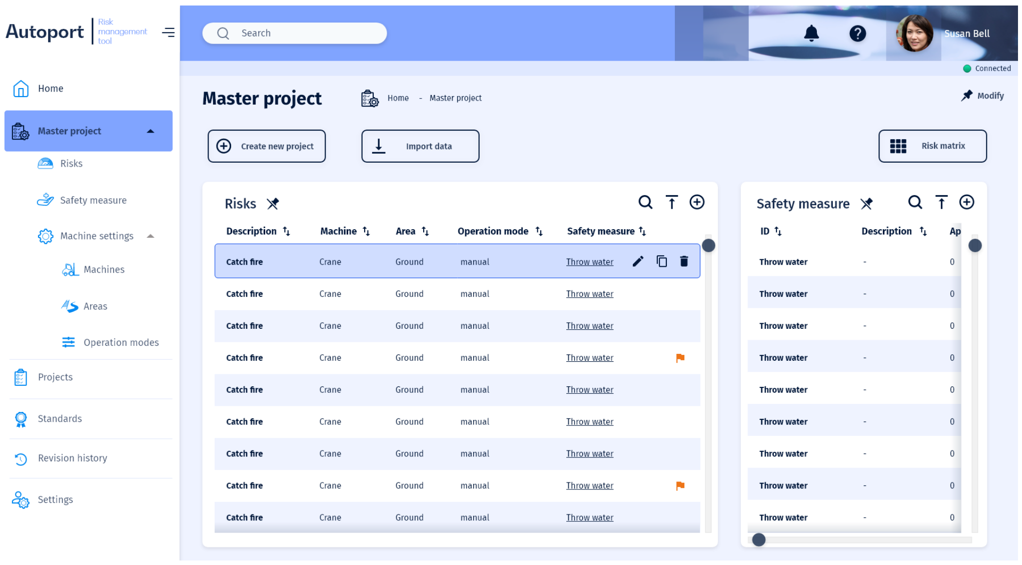
Front-end The front-end of the tool is also using C++, with Qt to implement the GUI.

3.2.2 Data modeling in the tool

The overall idea with the tool is to create a generic framework, which enables the modeling of different RE objects and their relations. In order to achieve that, object types and link types are the two fundamental data modeling units in the tool. Moreover, each object type can be configured by a number of parameters describing its attributes. With these three main building blocks most of the RE terms and relations can be modeled easily. The formulated structure is then filled with instances of the defined types.

Taking the data set used in the thesis as an example modeling problem, there would be three object types: customer requirements, software requirements and test specifications, with sufficient parameters for each type. The two link types would model the relationships between customer requirements and software requirements, and between software requirements and test specifications. After that the database would be populated with instances for each object type and where the object types are linked, the relevant link instances would also be created.

An example project set up of the above described data modeling is shown below:



Risks

Description	Machine	Area	Operation mode	Safety measure
Catch fire	Crane	Ground	manual	Throw water
Catch fire	Crane	Ground	manual	Throw water
Catch fire	Crane	Ground	manual	Throw water
Catch fire	Crane	Ground	manual	Throw water
Catch fire	Crane	Ground	manual	Throw water
Catch fire	Crane	Ground	manual	Throw water
Catch fire	Crane	Ground	manual	Throw water
Catch fire	Crane	Ground	manual	Throw water
Catch fire	Crane	Ground	manual	Throw water

Safety measure

ID	Description	Ap
Throw water	-	0
Throw water	-	0
Throw water	-	0
Throw water	-	0
Throw water	-	0
Throw water	-	0
Throw water	-	0
Throw water	-	0
Throw water	-	0

Figure 5: Autoport tool example GUI view

In this particular view, the overview of an operational hazard analysis is visible. The two main objects types here are risks and safety measures, which are linked to each other in a many-to-many relation. The developed IR system could help the linking between these to object types.

3.3 Information retrieval system architecture

The details of the prototype linking system's architecture are shown below.

3.3.1 Python prototype

In order to develop a proof of concept as fast as possible, in the initial phase of the project a separate Python[28]-based IR system was developed. This systems works separately from the previously introduced application on an real-life example project's data set.

3.3.2 Applied packages

NumPy [29] is an essential Python package used in computation-intensive projects. Using NumPy makes handling of N-dimensional arrays, similarly to MATLAB, efficient both in terms of development and execution time.

pandas [30] is a Python package created to handle .xls(x) imports efficiently. It was used for parsing the example project's set of requirements.

jsonpickle [31] is a tool which enables developers to save Python objects as json files. The parsed and pre-processed requirements and test specifications were saved via jsonpickle.

scikit-learn [32] is a free and accessible tool for machine learning related tasks in Python. During the project multiple classes of the package were used, for example sklearn metrics for defining cosine similarity among vectors and sklearn feature extraction for creating the tf-idf of the document corpus.

Matplotlib [33] was used during the project to plot all kinds of related data onto graphs.

Rank-BM25 [34] is a Python implementation of the popular BM25 ranking function, which is a probabilistic bag-of-words IR algorithm.

PySide [35] is Qt's Python implementation. It is now ported by the Qt company, thus it enables developers to create GUI-s using the Qt5 framework.

3.3.3 Data pipeline

The system architecture is best represented through its data pipeline. The figure below shows all the main modules which are responsible for a given step along the data flow. The five blocks in the upper stripe are representing the necessary steps to build a vector space representation of the example requirement data set. This part is done only once for the prototype. Naturally, once the system is integrated to the Autoport tool, some parts of this database populating phase is repeated once in a while, when a new object or link is added or an existing one is modified. The lower stripe shows the steps responsible for retrieving and answer for a given query on the previously constructed data set. This part is executed every time a new query arrives to the system.

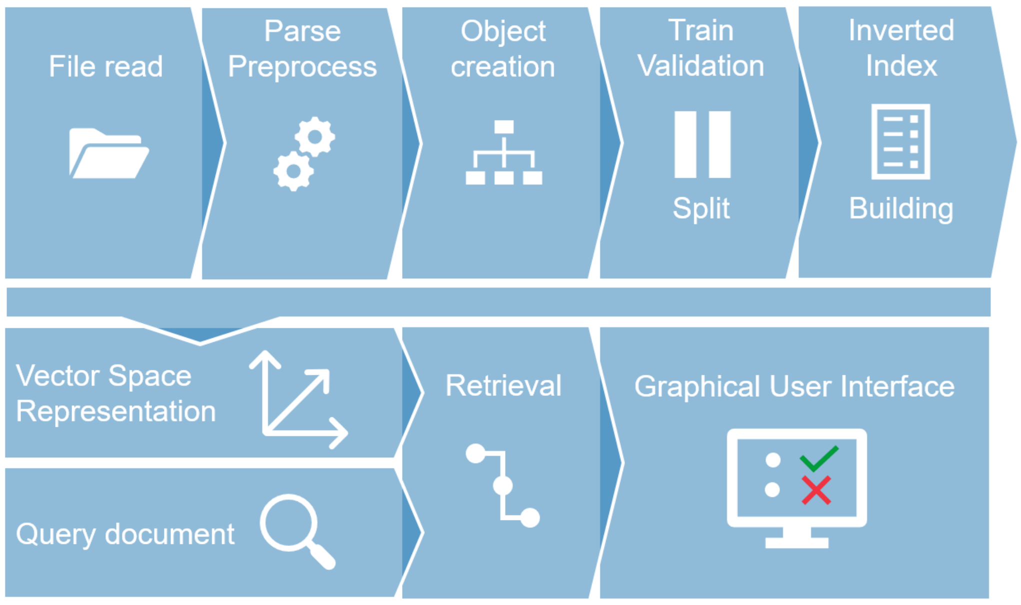


Figure 6: Prototype information retrieval system data flow

File reading The example data set arrived in to main file formats: the requirements in the form of excel sheets, while the test specifications were stored as comments in the beginning of the actual java test codes. The first step in order to build the inverted index and the vector space representation is to read these files.

Parsing and preprocessing Once the files were read into the memory, the algorithm parsed them line by line, detecting the relevant parts and throwing away the unnecessary rest. At this phase, besides selecting the useful parts, preprocessing also meant the slicing of free text parts into logically coherent units.

Object creation Once the relevant parts are identified, the next step is to organize the data in a structured manner. In the current implementation, it means the object instances and links are formed out of the unstructured data.

Train validation split In order to train the data on one part of the data set, and make unbiased estimated about the performance on another a train validation spilt is necessary. The key approach here was to assign a random weight for each link and based on that determine its set. It is further detailed in the experiments chapter.

Inverted index building The next step is to build the inverted index of the corpus using the scikit-learn implementation. The building process consists of collecting the dictionary of the corpus and representing each document by their term frequency vector, or counts vector.

Vector space representation The last step to create the vector space representation is the calculation of the tf-idf scores of the documents. The resulting vector representation has still the same dimensionality as the counts vector, the same as the length of the dictionary, but with the incorporation of inverse document frequency, the coordinates of the vectors are more representative for the importance of the terms.

Query document Normally, in an information retrieval system the query would be a free text search expression transformed into the vector space of the corpus. In the current case, the queries are actually already existing documents in the data set.

Retrieval The actual information retrieval steps only means the calculation of the distance metric between the query document and all possible other documents in the corpus. Then the top n documents, having the the closest vector representation to the query object based on the selected distance metric, are returned. The choice for the number of retrieved objects n can strongly influence the overall performance of the IR system.

Graphical User Interface In the final step, the retrieved shortlist of documents are shown in the prototype GUI, along with their confidence score (cosine similarity value) and a few key statistics about the retrieval.

3.3.4 Hyperparameters

The most important hyperparameters, which were tuned on the training set, are summarized here shortly.

Stop words Boolean parameter, which determines the usage of stop words list, provided by the scikit-learn Python package. The list contains a selection of common English stop words.

K-grams Two integer parameters, to define the minimum and maximum k-gram length, where the maximum is greater or equal to the minimum length. If the minimum and maximum are both defined as one, the dictionary simply contains the words, for higher values it will contain actual k-grams.

Distance/similarity metrics In order to find the closest vector to a given query vector in the vector space representation, a distance (or similarity) metric is needed. In the related work part several possible distance metrics were listed. In practice the L-norms were performing so badly compared to the cosine and dot product similarity, that in the end they were only tried on the fist few setups.

Inclusion of different object properties Two boolean parameters to control the inclusion of the test specifications' purpose and description properties. The full version, could implement a zoning technique described in the related work part, but the scikit-learn implementation was only prepared for binary inclusion instead of a sophisticated weighting.

Tf-idf or counts vector Binary parameter, which controls if the vector space representation is using tf-idf or it is a based on a simple counts vector (tf).

Tf-idf tuning: smooth idf, sublinear tf, normalization These three hyper-parameters are provided by the sckit learn tf-idf implementation, the first two are boolean values, while the last one is a binary selection. The smoothing adds a virtual document to the corpus, which contains each term of the dictionary. This prevents zero division. The sublinear tf counts with $1 + log(tf)$ instead of the simple term frequency. The normalization applied on the vectors during the build phase can either be according to the $l1$ or $l2$ norm.

Number of returned elements Integer parameter to control the length of short-list provided by the information retrieval system.

Similarity threshold Integer parameter as an alternative to the fixed length lists. It controls the similarity threshold, which decides if an object is retrieved or not.

3.4 Reasoning behind the selected information retrieval model

The final selected model for the retrieval was a scikit-learn implementation of a tf-idf based vector space representation. This section explains the most important factors which were taken into consideration and resulted in the selected model.

3.4.1 Sufficient performance for demonstration purposes

In order to demonstrate the business concept of the prototype, the information retrieval system has to reach an acceptable performance level. This can be considered as one of the most important criteria for the system under test. Without setting a strict threshold for it, it is easy to see, that the system has to strongly outperform a random method, which is clearly the case.

From the models with comparable complexity, the scikit-learn tf-idf clearly outperformed the slightly more rudimentary counts vector, while the selected BM25 implementation was less customizable, resulting in a somewhat worse performance compared to the tf-idf with hyperparameter tuning. See this at the Results section of the Experiment chapter.

3.4.2 Relatively short documents in the data set

The second consideration comes from the fact, that a typical requirement specification, thus the example data set too, contains relatively short documents. The short contiguous textual parts are a limiting factor on a system's potential performance. Therefore, it is an substantiated argument, that even significantly more complex IR systems could not necessarily outperform by big margin a simpler solution, such as the chosen tf-idf representation.

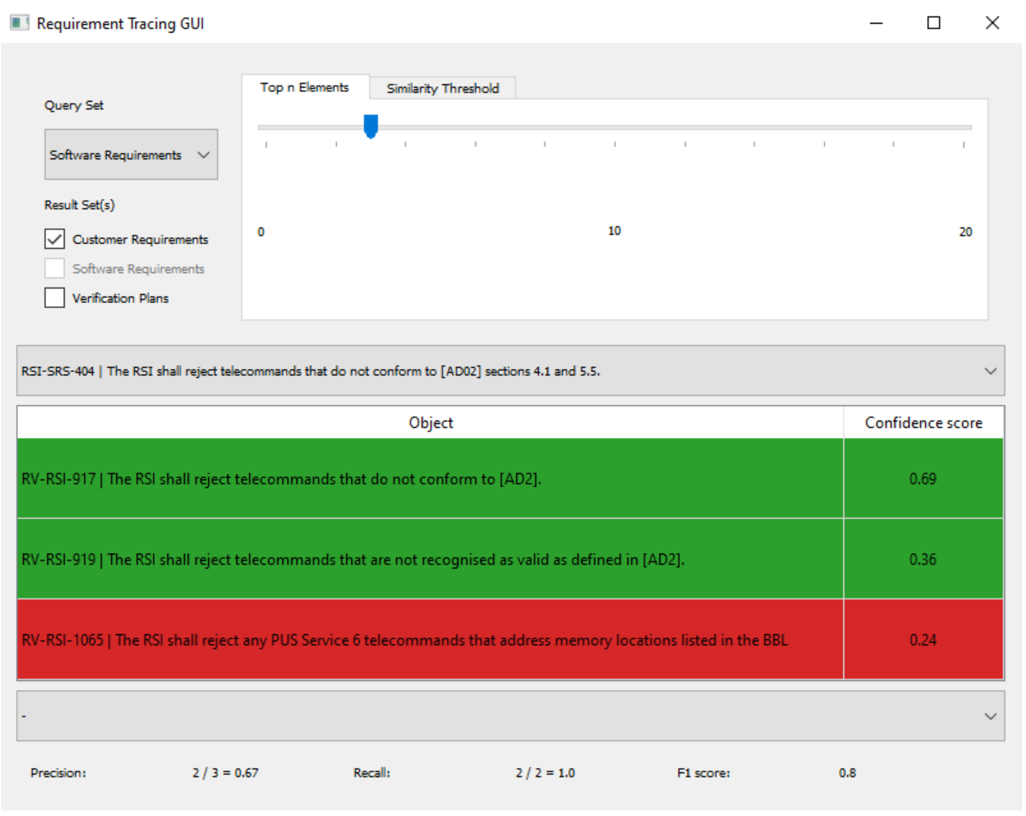
3.4.3 Time as a constraint within the project

As in almost any other case, the project's budget was a limiting factor on the time spent on the prototype implementation. Therefore, considering that the main purpose of the prototype was the demonstration of the business idea, less complex models were preferred, due to the shorter implementation time.

Considering the above mentioned criteria and limitations, the tf-idf is a justified model selection for the given task.

3.5 Example view of the demo IR tool

In order to easily demonstrate the functionality of the IR system, a demo GUI was developed, which shows the role of the system as well as the effect of the most important system parameters and some basic performance statistics.


Requirement Tracing GUI

Query Set: Software Requirements

Result Set(s):
- [✓] Customer Requirements
- [ ] Software Requirements
- [ ] Verification Plans

Top n Elements | Similarity Threshold

0 10 20

RSI-SRS-404 | The RSI shall reject telecommands that do not conform to [AD02] sections 4.1 and 5.5.

Object	Confidence score
RV-RSI-917 | The RSI shall reject telecommands that do not conform to [AD2].	0.69
RV-RSI-919 | The RSI shall reject telecommands that are not recognised as valid as defined in [AD2].	0.36
RV-RSI-1065 | The RSI shall reject any PUS Service 6 telecommands that address memory locations listed in the BBL	0.24

-

Precision: 2 / 3 = 0.67 Recall: 2 / 2 = 1.0 F1 score: 0.8

Figure 7: IR system demo GUI view

Based on the related functionality, the view is dived into three separate main units:

1. Control bar
2. Query selector
3. Return bar

The control bar lets the user try the effect of different parameters. It is further divided into two sections, the left part is for setting up the query and result sets, while the right part is to define the retrieval criteria. One can select in the tab widget if it should be always a fixed number of n elements or a varying number of objects based on their similarity to the query object. Within the tabs a slider lets the user choose the exact value of the parameter related to the retrieval mode.

The query selector is a simple drop-down menu, where one can select any arbitrary object from the query set. This will be the object, to which similar objects are tried to be retrieved from the result set.

The last unit is the return bar. This part is again divided into three parts, on the top there is a place for the retrieved, recommended objects, in the middle there is a drop-down selector with the same visual outlook as of the query drop-down's, while on the bottom there is a place for statistics related to the retrieval.

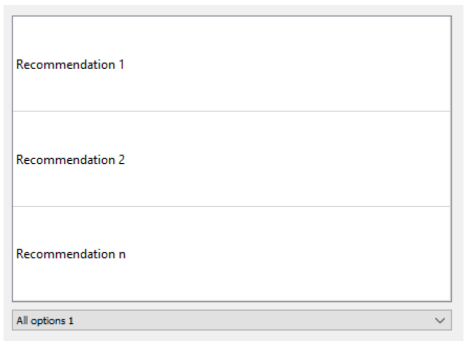
The retrieved objects are marked with green or red background color, based on if they are relevant retrievals (if in reality there is a link between the retrieved object and the query object in the data set). Besides the objects' id and the first hundred characters of their description, their cosine similarity (to the query object) is also shown as confidence score.

The result set's drop-down only serves as a presentation, how the objects would have to be traced without the recommendations.

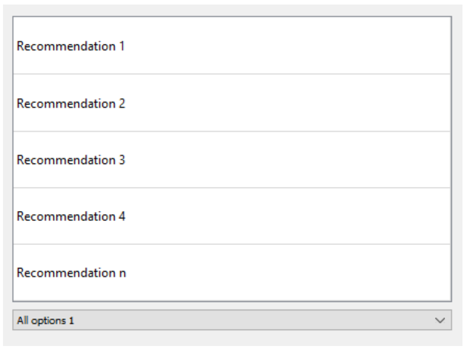
Finally, three basic statistic of the current retrieval are calculated and indicated on the bottom stripe: precision, recall and the F1 score.

3.6 Example view for tracing objects in the application

In the Autoport tool requirements, and other objects, are shown in a tabular form.Users can click on one of the button shown on the right hand side of the tables on hover. The click event on these buttons shows a popup window for editing, deleting or linking objects. Upon integrating the IR system to the tool, clicking on the linking button could evoke a dialog similar to the IR demo GUI, but naturally without the setup parameters, the statistics and the color coding. A possible realization of such a dialog is shown below:



With $n = 3$



With $n = 5$

Figure 8: Example GUI view for tracing objects

Based on the active row in the objects' table, so on the object to which other objects are to be traced, a query is formulated and passed onto the IR system. The resulting list of retrieved objects are presented then as it is visible on the figures.

The two sub figures show the scenarios for 3 or 5 retrieved objects. If none of the recommended objects were found to be relevant, the user has to select from the drop-down selector at the bottom of the dialog box.

3.7 Integration to the Autoport tool

As of the time of writing, the Autoport tool and the prototype Python IR system are working completely separately, as the main goal of the thesis was the demonstration of the feasibility of such approach. In the future however, if the Autoport tool is decided to apply such technology, the interfacing or integration is inevitable.

Interfacing would mean to keep the current code base of the prototype as Python, and in addition to implement a connection to the database of the Autoport tool and an API to send requests from the UI of the tool. A more thorough approach would be the complete re-implementation of the prototype's algorithm in the C++ tool, storing the necessary values, such as the dictionary and the vector representation of documents, within the existing database of the tool in additional tables.

The interfacing approach might be quicker to implement, but harder to maintain because of the separate code base. On top of that, a key requirement to the Autoport tool is its speed, and even tough during the tests with the prototype no performance issues were detected, it is certainly slower to call API's of a Python implementation, than to solve the problem within the main C++ application. For the mentioned reasons, the complete C++ re-implementation is suggested.

4 Experiments

4.1 Structure of example data set

The previously mentioned ExoMars project Software Requirement Specification and the related test specifications were used as example data set. The following three object types were identified and separated - the object types' features are also explained below:

Customer Requirements	**Software Requirements**	**Test Specifications**
id	id	id
requirement text	requirement text	purpose
link to software requirements(s)	link to customer requirements(s)	description
	link to test specification(s)	link to software requirements(s)

Table 1: Example data set structure

After parsing and pre-processing the data, which included the selection of relevant features and the elimination of links to unit tests, the example data set consists of 128 customer requirements, 280 software requirements and 87 test specifications.

The automated tracing can be run for customer requirement – software requirement or software requirement – test specification pairs in both directions.

The data structure is illustrated in the following graphs, each showing the instances of two object types and their linkage among them. It is visible that the customer requirements – software requirements relations are more distributed, most of them are just a small group of connected objects. On the other hand, software requirements – test specifications relations have one big connected component, involving most of the objects and only a modest share of the objects are in small groups.

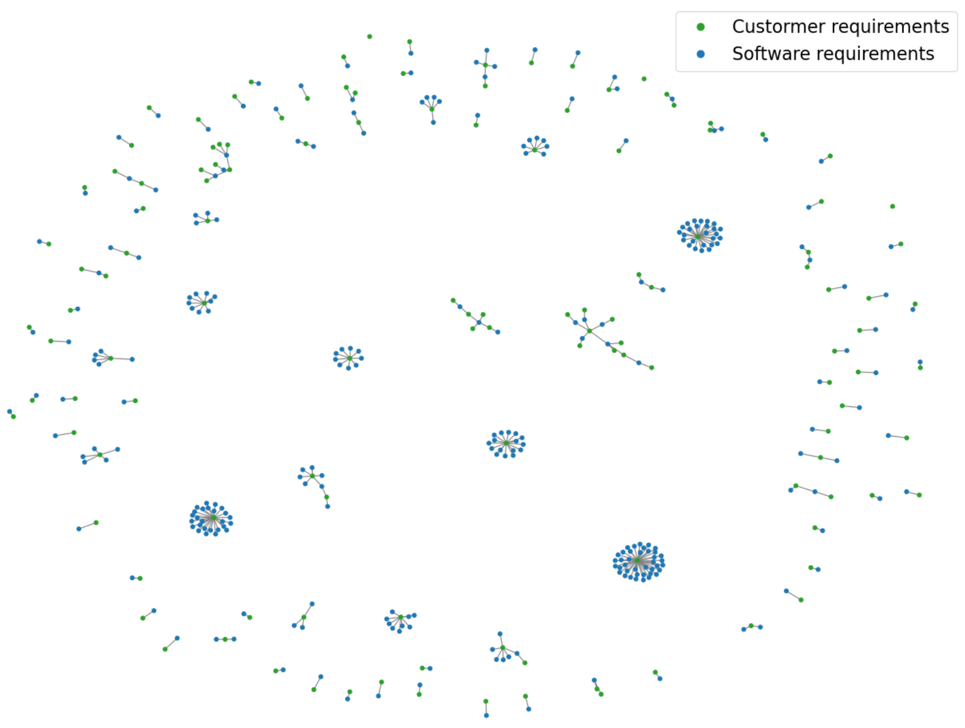


Figure 9: Customer requirements – Software requirements relationship illustration

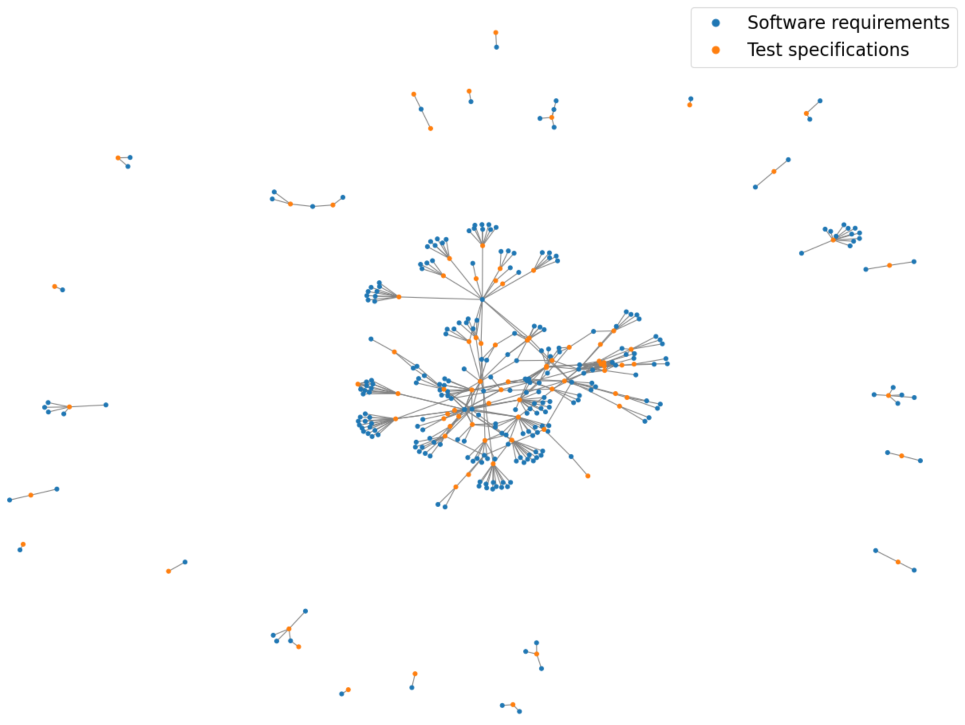


Figure 10: Software requirements – Test specifications relationship illustration

4.2 Definition of train and validation set

The prototype IR system's example data set needs to be divided into a train and validation set. As a standard step in machine learning-like problems, this allows to get an unbiased estimate of the performance of the algorithm under test.

When dividing the data set into train and validation set, it is important to distribute the objects homogeneously, otherwise the performance estimation is not going to be valid. Although the IR solution does not have a classical learning phase, it has several hyper parameters, which can strongly influence its performance. Therefore, the train-validation split is inevitable for the current thesis too, even thought the train set is only used for the hyper parameter selection.

4.2.1 Properties of ideal train-validation split

According to [36] the two most important criteria regarding the train-validation split of the data is, that

1. Both the train and validation set shall be large enough to allow statistically significant outcomes.

2. Both the train and validation set shall be representative to the original data set as a whole. The two sets shall not have different characteristics.

While splitting the data, these properties always have to be fulfilled.

4.2.2 Difficulties of train-validation split in the current project

In the case of this particular project, the division was especially challenging, as the objects of the data set are interconnected. The different types of objects, for instance customer and software requirements or software requirements and test specifications, are connected to each other in a way, that only objects of different types can be linked, therefore the data set is possible to represent as a bipartite graph.

Thus, one cannot divide the objects into training and validation set at any arbitrary point, without breaking some of the links. An attempt to overcome this challenge was proposed in the form of an algorithm for a selection - with breaking as few links as possible. Software requirement – test specification tracing is used as an example to describe the workflow of the algorithm.

4.2.3 Unsatisfactory first trial for train-validation split through object-based approach

The basic steps of the algorithm are described in the following:

1. Add an arbitrary software requirement to the involved requirements
2. Loop through its linked test specifications and add them to the involved tests
3. Switch back to the linked software requirements of the involved tests from the previous point and mark them as involved too
4. repeat between the second and thirds steps until there are no objects left (either because all were added or no more links are point towards the not included objects)

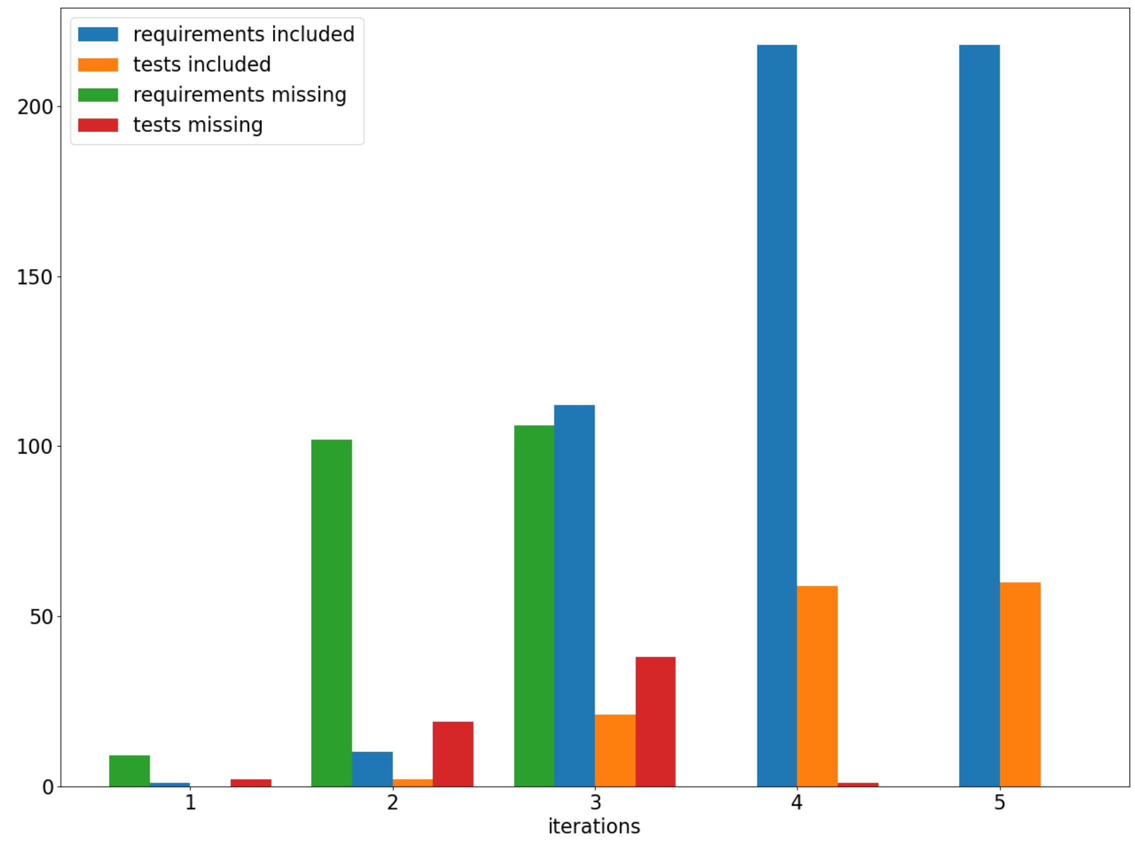


Figure 11: Intermediate results of the object-based approach for train-validation split

On the graph above you can see how the number of included and missing requirements and tests are changing over the iterations of the algorithm. After a few iterations the algorithm converges, and no more objects are added. 218 out of 280 (78%) software requirements and 60 out of 87 (69%) tests specifications are included, these serve as training set, and the remaining (missing) objects, 62 software requirements (22%) and 27 test specifications (31%) are forming the validation set.

The customer requirements are divided based on the previously shown division of the software requirements.

The graphs below show the resulting division of the objects into training and validation set in both cases.

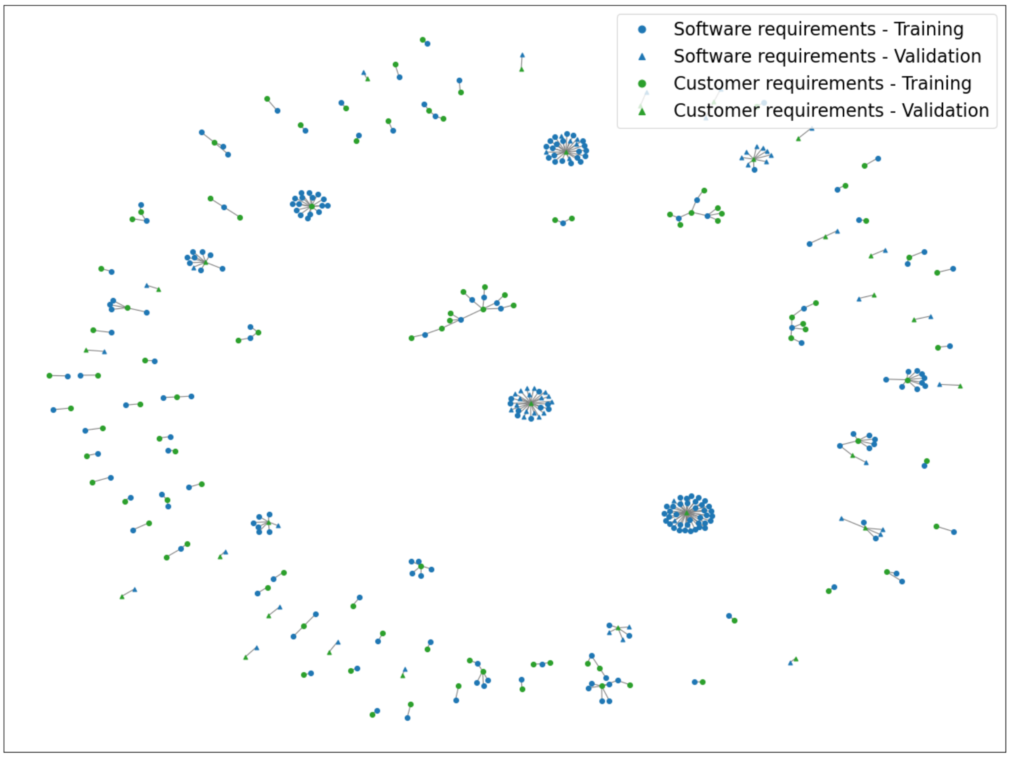

● Software requirements - Training
▲ Software requirements - Validation
● Customer requirements - Training
▲ Customer requirements - Validation

Figure 12: Object based division of Software requirements and Test specifications relationship illustration

As it is visible on the graph above, some of the nodes, i.e. the customer or software requirements, are linked to nodes from a different group of nodes. Thus to problem of linking in between groups - validation to training and vice versa - was not resolved for the customer requirements.

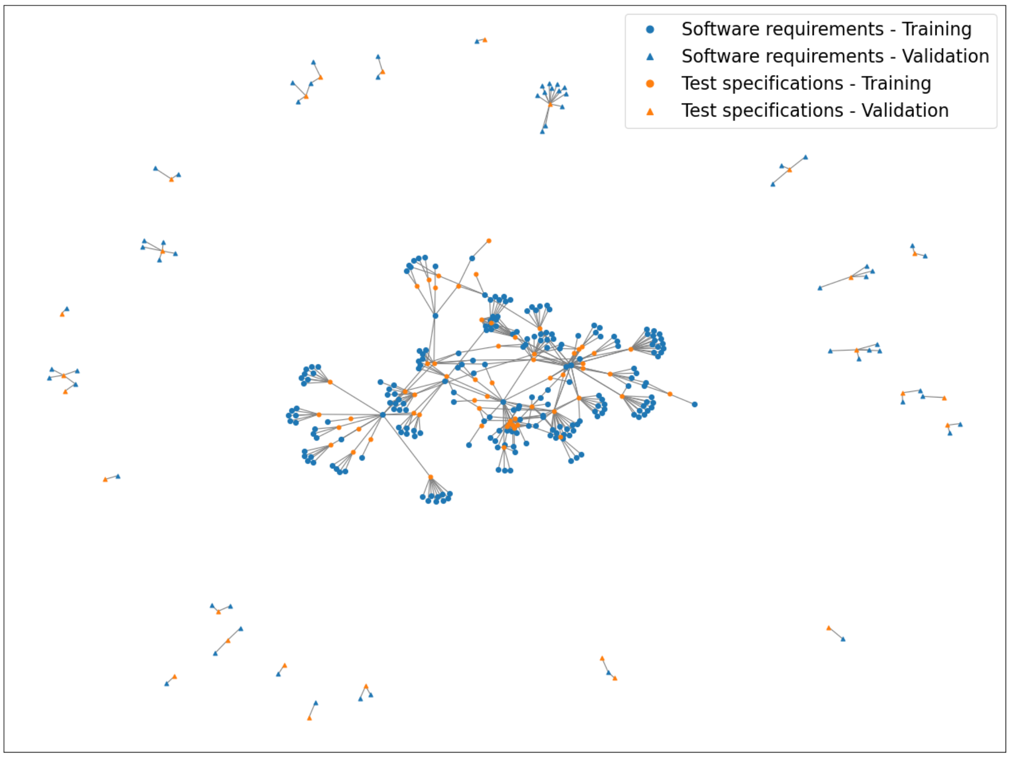


Figure 13: Object based division of Software requirements and Test specifications

For the test specification there is no inter-set linking, but it is easily visible, that the separation was violating the second desired property, i.e. the two sets should not have different characteristics, because all the small groups on the side of the figure belong to the validation set, while the big connected component serves as training set. Thus, a risk arise that the degree distributions in the train and validation set nodes are different, which is not desired, as the resulting evaluation measure can yield different results. This property of the division is further discussed at 4.2.5.

In order to handle the problems occurred another approach was developed.

4.2.4 A better idea: link-based approach for train-validation split

In order to overcome the inter-set linking, a change of paradigm was needed regarding the train validation split. Instead of dividing the objects, the vertices of the graph, into two, now the links, i.e. the edges of the graphs, are split. It is a better approach, as the IR system actually predicts links between objects and not the objects themselves. In addition, the degree distribution of the two sets are also more similar using the improved way.

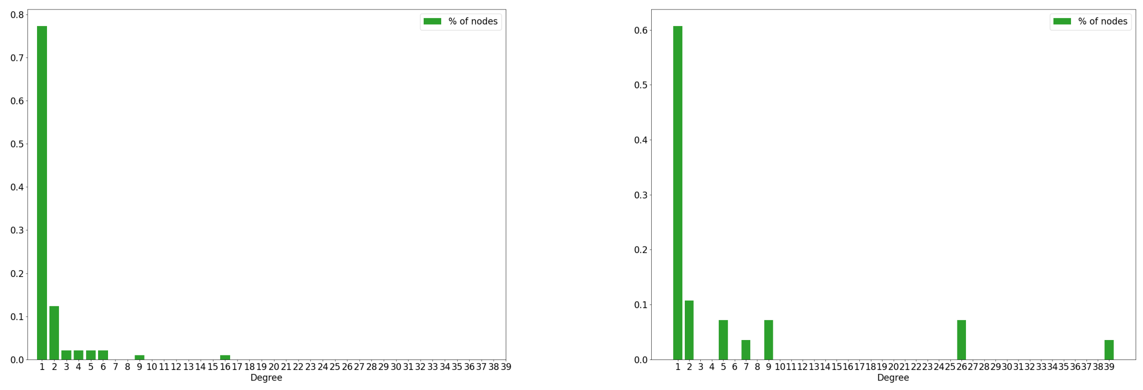
The basic steps of the link-based algorithm are described in the following:

1. List all existing links (grouped by object type)
2. Assign a weight to each link
3. Select top x percent as validation by a given threshold
4. Store if it is training or validation for each link
5. When querying, only take into account the links matching the query phase (training or validation)

4.2.5 Comparison of degree distributions

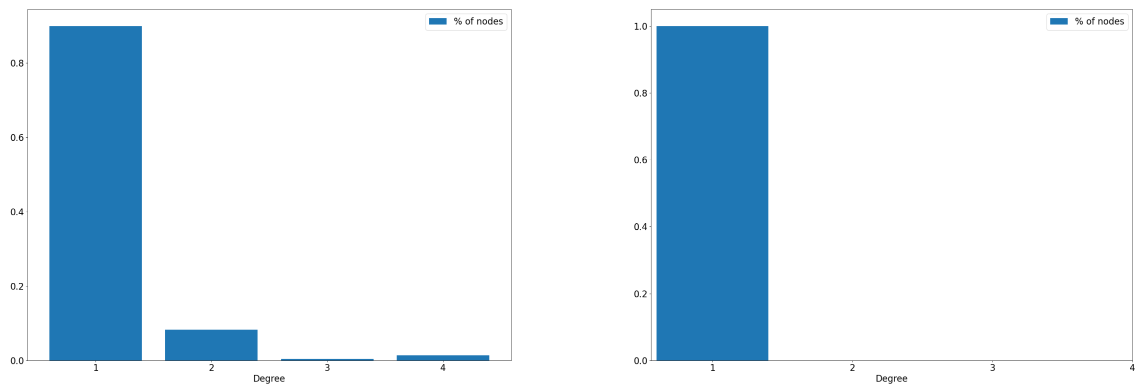
In order to compare the performance level of the object and link based division approaches a graphical comparison was applied, which shows the degree distributions of the train and validation splits.

Three distinct versions are shown, first the results of the unsatisfactory object based approach, then two instances made by the link based approach: an 80-20 train - validation division, which is later on used by the model and a 50-50 train - validation division to show a more optimal case.



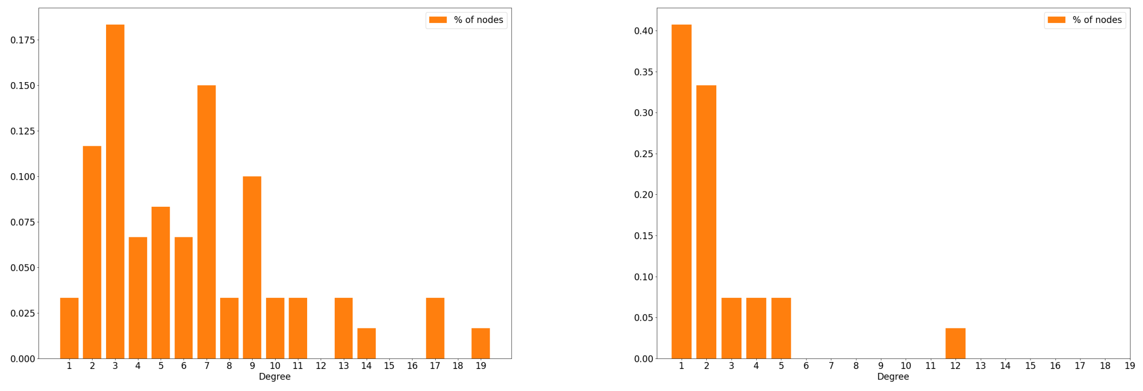
Training set Validation set

Figure 14: Customer requirement objects' degree distribution for object based division



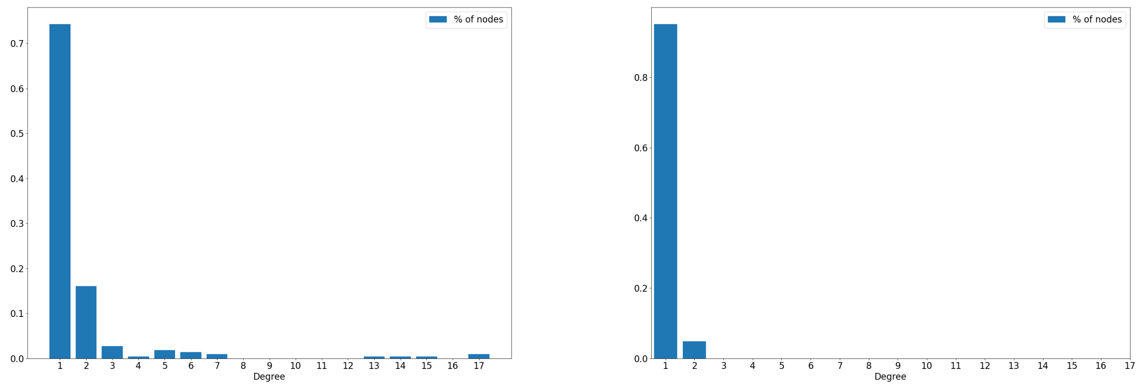
Training set Validation set

Figure 15: Software requirement objects' degree distribution (linked to customer requirements) for object based division



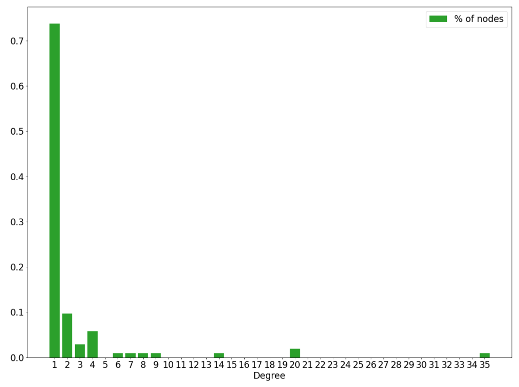
Training set Validation set

Figure 16: Test specification objects' degree distribution for object based division

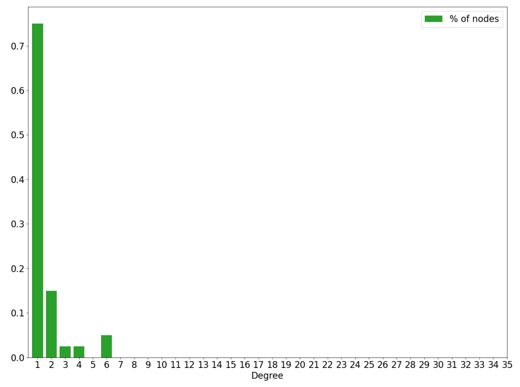


Training set Validation set

Figure 17: Software requirement objects' degree distribution (linked to test specifications) for object based division

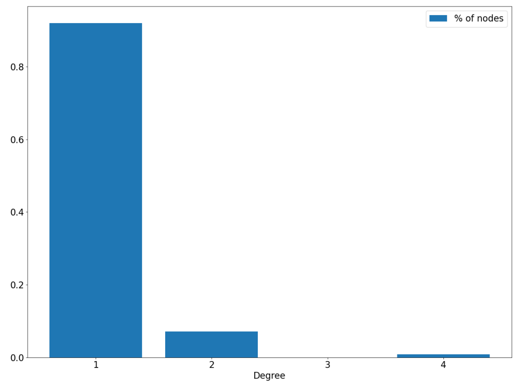


Training set

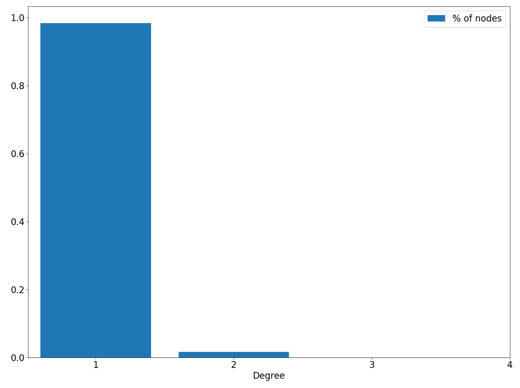


Validation set

Figure 18: Customer requirement objects' degree distribution for 80-20 link based division

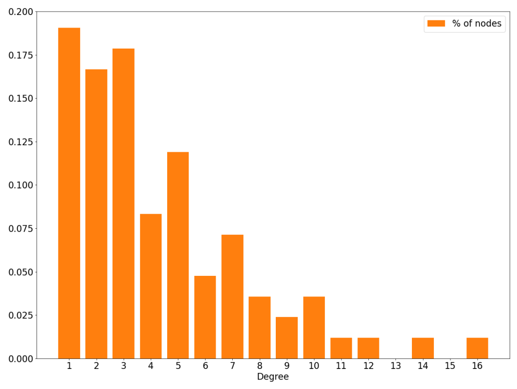


Training set

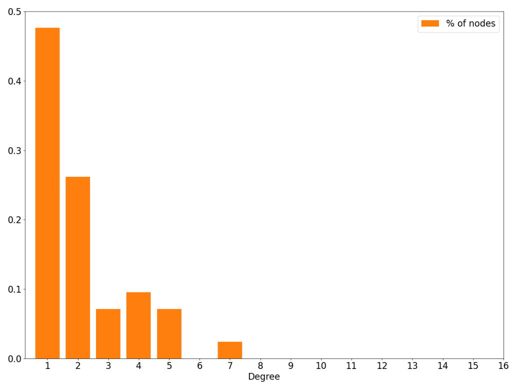


Validation set

Figure 19: Software requirement objects' degree distribution (linked to customer requirements) for 80-20 link based division

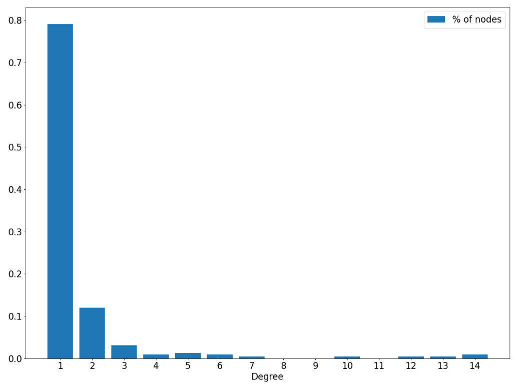


Training set

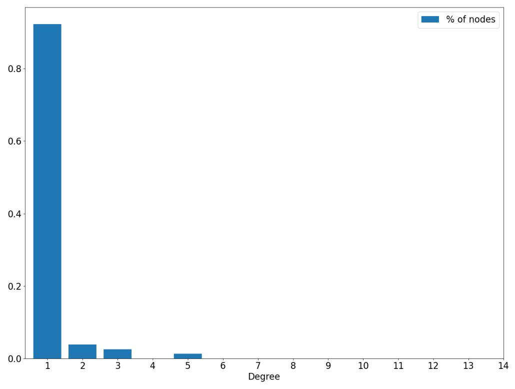


Validation set

Figure 20: Test specification objects' degree distribution for 80-20 link based division

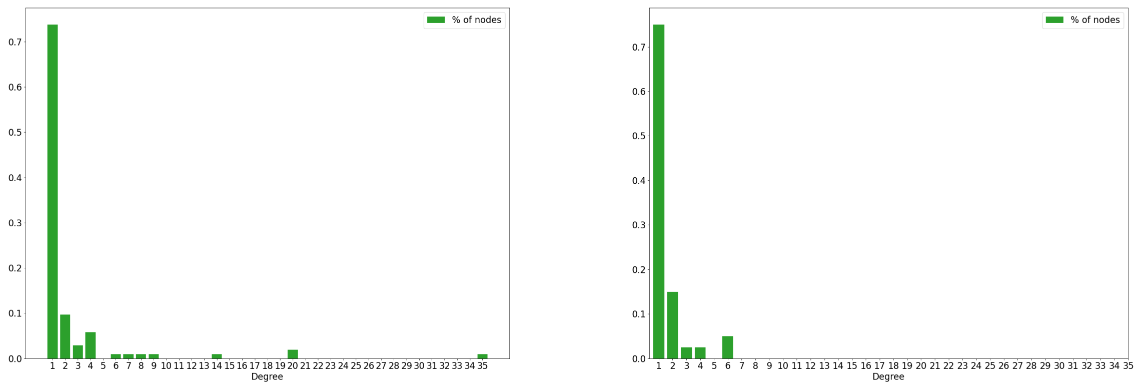


Training set



Validation set

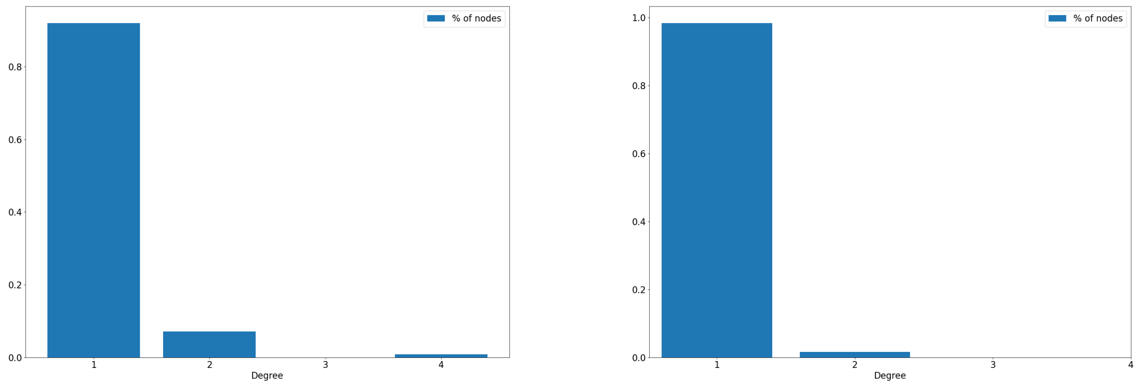
Figure 21: Software requirement objects' degree distribution (linked to test specifications) for 80-20 link based division



Training set

Validation set

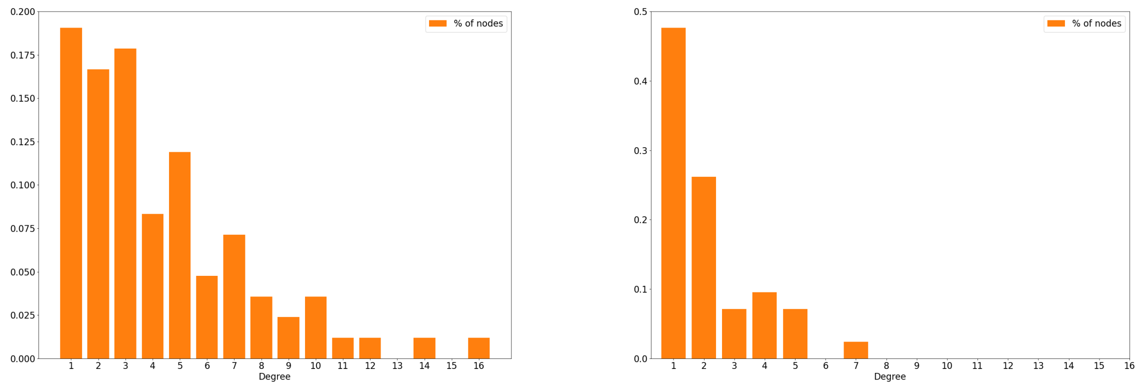
Figure 22: Customer requirement objects' degree distribution for 50-50 link based division



Training set

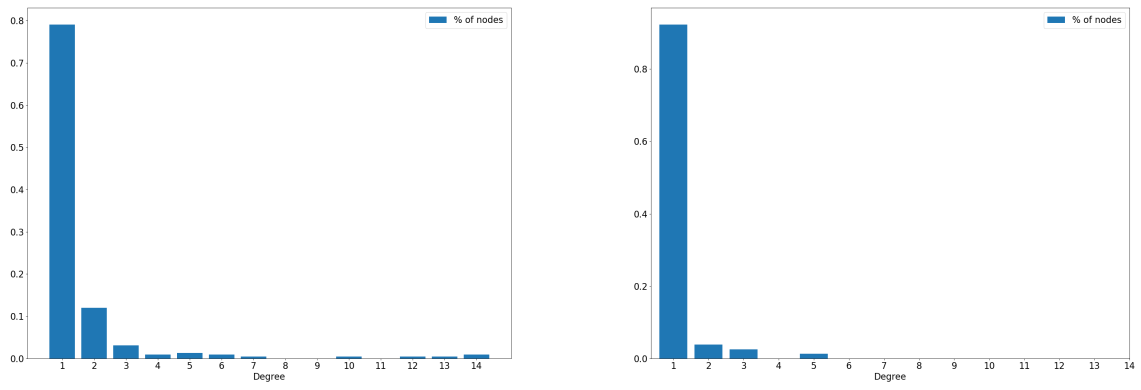
Validation set

Figure 23: Software requirement objects' degree distribution (linked to customer requirements) for 50-50 link based division



Training set Validation set

Figure 24: Test specification objects' degree distribution for 50-50 link based division



Training set Validation set

Figure 25: Software requirement objects' degree distribution (linked to test specifications) for 50-50 link based division

It is visible on the graphical comparison of the degree distributions, that the linked based approaches create more similar degree distributions in the train and validation sets, then the object based approach. The drawbacks of the object based approach are most visible on the respective test specification charts, where the train and validation set distributions are sharply discrepant. This was shown before just by looking at the plot of the graph made out of the objects and their links, where in some cases - by the object based approach - all the high degree nodes belonged to the training set.

The link based approach is also not perfect, on the graphs it is visible that there are differences in the degree distribution of the two sets, especially in the higher degree nodes, as there are fewer of them, and the random weighting handles all links the same way, no matter the their node's degree. The 50-50 division naturally outperforms the 80-20 as it is allows a more similar division of the sets. If the sets degree distributions are aimed to be even more similar, a more sophisticated weighting approach, one that takes the node's degree into account, should be applied.

4.3 Evaluation measures

The evaluation of any machine learning-like system, in our case an IR system, is in some sense the last step of creating the pipeline, nevertheless it is inevitable in order to be able to select the most efficient IR system as well as to predict its performance on future -unseen- data.

Regression and classification problems are evaluated differently, and IR problems in general can be interpreted as a binary classification into two classes: relevant and irrelevant documents. Thus, one of the fundamental tools to evaluate classification problems, the confusion matrix, is a suitable starting point.

4.3.1 Confusion Matrix

Based on [37], we can define the confusion matrix in the following way:

		Actual Class	
		Condition Positive CP	**Condition Negative CN**
Predicted Class	**Predicted Positive PP**	**True Positive TP**	**False Positive FP**
	Predicted Negative PN	**False Negative FN**	**True Negative TN**

Table 2: Confusion matrix

The confusion matrix divides the data along two criteria, each elements actual class and their predicted class. Based on these criteria, each prediction falls into one of these four categories:

- **True Positive - TP:** both the actual and the predicted class is positive, thus it is a correct prediction. In IR context the element is relevant and it was retrieved.

- **False Positive - FP:** the actual class of the element is positive but the predicted class is positive, thus it is an incorrect prediction. In IR context the element is non-relevant but it was retrieved. It is also called Type I error.

- **False Negative - FN:** the actual class is positive but the predicted class is negative, thus it is an incorrect prediction. In IR context the element is relevant but it was not retrieved. It is also known as Type II error.

- **True Negative - TN:** both the actual and the predicted class is negative, thus it is a correct prediction. In IR context the element is non-relevant and it was not retrieved.

The following equations capture the relations between the headers of the matrix to the cells of it:

$$PP = TP + FP \tag{10}$$

$$PN = FN + TN \tag{11}$$

$$CP = TP + FN \tag{12}$$

$$CN = FP + FN \tag{13}$$

4.3.2 Accuracy

Accuracy measures the fraction of correct predictions among all predictions:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{14}$$

The biggest limitation of accuracy is that treats the positive and negative classes equally, which can be problematic for two reasons. Firstly, if the data is unbalanced - i.e. that one of the classes has much higher number of instances than the other - then the value of accuracy will depend much more on the class with more instances, thus it might not be right to treat the two classes equally. Secondly, if the cost of a false negative and the cost of a false positive is excessively different, then it is also not correct to to handle the two classes in the same way.

The following measures try to overcome these weaknesses of the accuracy.

4.3.3 Precision, recall

Precision (or positive predictive value, PPV) is the fraction relevant elements among all retrieved elements:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{PP} \tag{15}$$

It is easy to see, that it tries to capture how precise was the retrieval, how many of the retrieved objects were actually relevant. If the cost of false positives is relatively high, then precision can be a good measure to evaluate the models performance.

Recall (or sensitivity) is the fraction of retrieved elements among all relevant elements:

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{CP} \tag{16}$$

This measure, on the other hand, is showing how complete was the retrieval, how many of all relevant objects were returned. Therefore, it is useful when the cost of false negatives is high.

Generally speaking, if the threshold between positive and negative predictions is lowered - i.e. more elements are predicted to be positive - the recall grows, while the precision drops. The same way, if the threshold is increased - i.e. less elements are predicted to be positive - the recall drops, while the precision increases.

Therefore, there is a trade-off between precision and recall in all classification problems and the correct choice of the threshold can depend on the application. Because of this, a more complex measure is needed in order to evaluate a system's performance by a single number.

4.3.4 F1 score

In order to balance the trade-off between precision and recall, the F1 score[38] is introduced in the following way:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{17}$$

It is the harmonic mean of the previous the previously introduced two measures. As none of the precision or recall takes into account the true negatives, their value also does not influence the F1 score either. This property is generally not a problem in case of IR systems as usually only a handful of documents are relevant.

4.3.5 Matthews correlation coefficient

The Matthews correlation coefficient[39] (MCC) is the most complex one among the listed measures. it is calculated in the following way:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{18}$$

Essentially, it is a correlation between the actual and the predicted classifications. It takes into account all 4 quadrants of the correlation matrix, in contrast to the F1 score even the true negatives, which ensures a balanced measure. This means, that it can be useful in case of classification of highly unbalanced classes too.

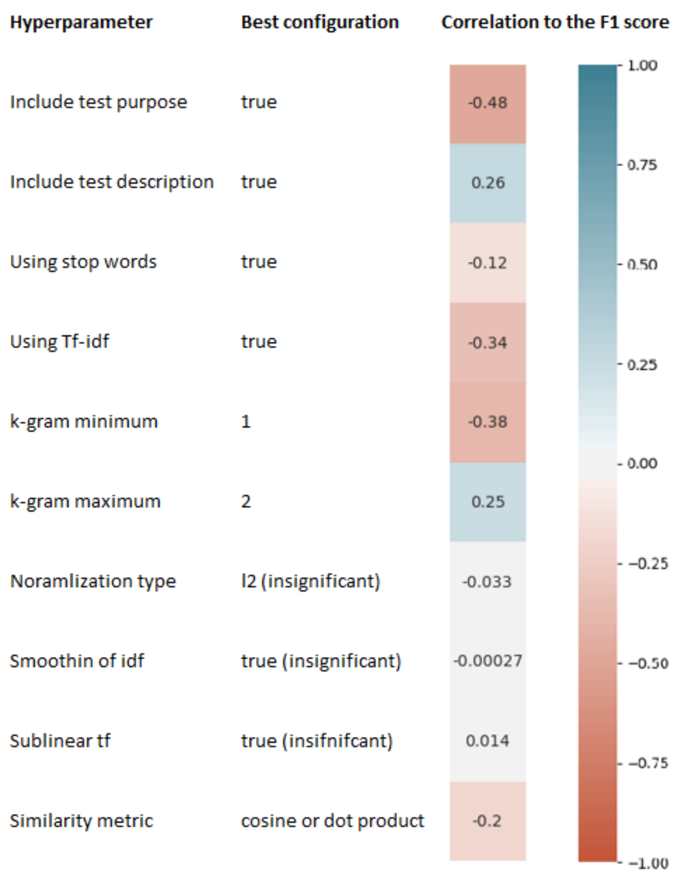
Another property of the MCC is its symmetry. This means, that the measure's value does not depend on the labeling of the classes.

4.4 Hyperparameter tuning with grid search

The hyperparameters defined in the system architecture chapter, were optimized using the grid search method. The grid search method basically goes trough the gird of all possible hyperparameter configurations with a given resolution. In the current case, most of the parameters were boolean, meaning only two states in the grid search, and the remaining non-boolean variables were also discrete, therefore there was no need for discretization.

As a result, the optimization was carried out on 2880 discrete states (not counting the states for the number of returned objects, detailed in the next section), which is still small enough to do it with the grid search method. If the hyperparameter space were significantly greater, containing several continuous dimensions too, a more suitable tool could be the random search method, where the discrete states are not defined along the grid with equal step size, but randomly. This later method can speed-up the optimization process, but as mentioned before, it was not needed in the current project.

The results of the optimization are visible on the figure below:



Hyperparameter	Best configuration	Correlation to the F1 score
Include test purpose	true	-0.48
Include test description	true	0.26
Using stop words	true	-0.12
Using Tf-idf	true	-0.34
k-gram minimum	1	-0.38
k-gram maximum	2	0.25
Noramlization type	l2 (insignificant)	-0.033
Smoothin of idf	true (insignificant)	-0.00027
Sublinear tf	true (insifnifcant)	0.014
Similarity metric	cosine or dot product	-0.2

Figure 26: Results of hyperparameter tuning

The first column shows the analyzed hyperparameters, the seconds column presents their value in the optimum, while the third column shows each parameters' correlation to the F1 score. The correlation was calculated using the Pearson correlation coefficient (PCC). The PCC yields results on the $[-1, +1]$ scale. The magnitude of the PCC shows the strength of the linear correlation, while the direction shows if it is a positive or negative association. In the current case, the direction is only applicable for the k-gram tuning, as for all the other values the numeric value assignment to the parameter was arbitrary, therefore only the magnitude carries useful information in these cases.

Looking at the concrete results, the most important hyperparameters are the inclusion of the purpose fields in the test specifications (true better), the usage of tf-idf over simple tf representation (true better) and the k-gram minimum value (1 best). The test descriptions (true better) and the k-gram maximum value (2 best) are still important, as well as the similarity metrics (cosine similarity is better in some cases, but in the best configuration its irrelevant). The usage of stop words have a smaller impact (true better), and the remaining parameters have practically no effect on the performance of the information retrieval system.

4.5 Number of returned objects

One of the most important tasks in the project is to determine the optimal number of returned objects. This number n will be the length of the shortlist, similarly as it is shown in 3.6. In this thesis, two different approaches are shown to determine n, the first one is a more traditional machine-learning-like optimization, while the second one uses tools from computation user interface design optimization.

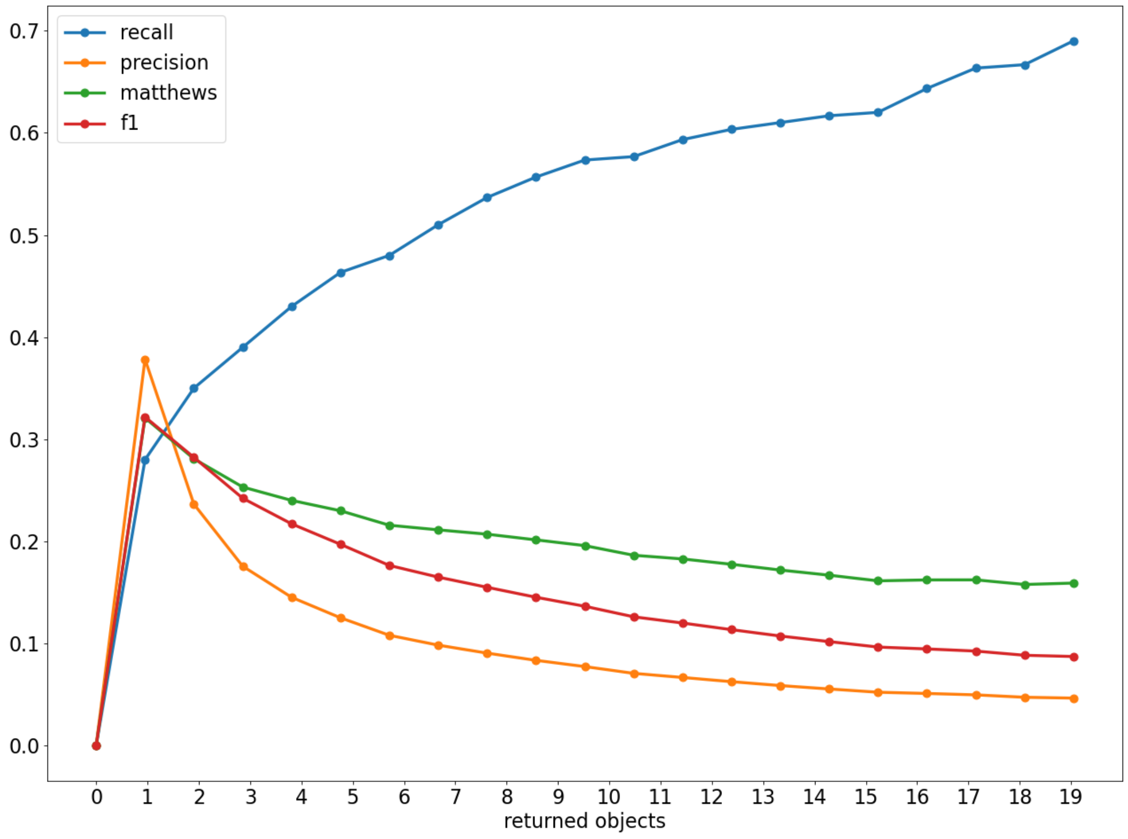


Figure 27: Recall, Precision, F1 and MCC on 80-20 division training set

4.5.1 Hyperparameter optimization based on MCC

This approach treats the number of returned objects simply as a hyperparemeter and chooses the setup, which performs best on the training set. There are a number of ways to define best performance, three of them are mentioned here:

1. Hyperparameter-selection yielding highest MCC

2. Hyperparameter-selection yielding highest F1 score
3. Hyperparameter-selection at the intersection of the precision and recall.

Both the MCC and the F1 score provides the highest value for a single returned object. It is so, because the precision drops sharply after the first returned element, while recall grows steadily but slowly.

The precision – recall intersection is between one and two, which fortifies the parameter choice to be one.

4.5.2 Alternative optimization based on Fitts's law

The down-side of the previously described single evaluation metric approach is that it optimizes the shortlist purely based on the retrieval, while in reality we can add more information to it. We know that the shortlist will be shown as a specific UI element, therefore it would make sense to incorporate the information originating from this fact.

The field concerned with these kind of questions is the computational user interface design, applying its laws and priciples can lead the the emergence of an alternative optimization method for the number of elements in the shortlist. One example is the Fitts's law, which was used multiple times to optimize menu layouts. Below, an approach is introduced as a basis for the incorporation of data from computation UI design.

The following approach is applied in order to determine the expected time of selecting a relevant requirement on a user interface similar to the one on Figure 8:

$$\mathbb{E}[T] = \mathbb{E}[T_n] + \mathbb{E}[T_\emptyset] \tag{19}$$

where:

- $\mathbb{E}[T]$ - Expected time to select a relevant element
- $\mathbb{E}[T_n]$ - Expected time to select from the recommendation shortlist with n elements
- $\mathbb{E}[T_\emptyset]$ - Expected time to select from the list of all elements

The addends of the equation (19) are further detailed below:

$$\mathbb{E}[T_n] = \sum_{i=1}^{n} P(i) \cdot T_n(i) \tag{20}$$

where:

- $P(i)$ is the probability that the i^{th} element is relevant. It is derived from the same data, which was used to create Figure 27
- $T_n(i)$ is the expected time needed to select the i^{th} element - using Fitts's law

$$T_n(i) = a + b \cdot log_2 \left(\frac{(i-1)W_n + \frac{W_n}{2}}{W_n} + 1 \right) = a + b \cdot log_2 \left((i-1) + 1.5 \right) \tag{21}$$

where:

- a and b are Fitts's law's constants
- W_n is the width of a list element, if the list contains n elements

$$W_n = \frac{W_{max}}{n} \tag{22}$$

where:

- W_{max} is the maximal width of a list element - the size of the place reserved for the list

$$\mathbb{E}[T_\emptyset] = P_\emptyset \cdot T_\emptyset = (1 - \sum_{i=1}^{n} P(i)) \cdot T_\emptyset \tag{23}$$

where:

- $P_\emptyset$ is the probability that none of the shortlisted elements were relevant
- $T_\emptyset$ is the expected time needed to select from all m objects below the shortlist

$$T_\emptyset = const \tag{24}$$

To sum it up, the first addend in (19) sums over the shortlist, and calculates the product of the probability that the i^{th} element is relevant by the expected time needed to select the i^{th} element. At the same time, the second addend in (19) simply multiplies the probability that none of the shortlisted elements were relevant by the expected time needed to select from all m objects.

As a result, for each n the the expected selection time is calculated based on (19), and the one yielding the lowest number is selected as optimal shortlist length.

Remark: the model shown here is missing some important aspects, and therefore in the current form it is not sufficient to fulfil its intended purpose, so to optimize the length n.

The most important deficiency is to take care of the time of decision making. The Fitts's law is simply a pointing model, so in the current case it is assumed, that a user can decide immediately about the relevance of a given recommendation. This is obviously not sufficient this way.

Another missing point is the actual value for the constant parameters, e.g. in the Fitts's law as well as for the selection time from the remainder list.

In order to effectively use the proposed approach, the above mentioned deficiencies are to be mitigated.

4.6 Returning elements based on similarity threshold

Instead of setting a fixed number of n returned elements - either based on MCC or on Fitts's law - an alternative way to determine the length of the list of the retrieved elements is to define a similarity threshold. Obviously, objects with a similarity to the query object smaller than the threshold, would not be retrieved, while if the similarity is above, the object will be part of the return set.

This approach could be beneficial, as it would make the retrieval lass sensitive to the variation in the number of links the objects have in the data set, i.e. in the variation of the degree of the vertices on Figure 9 and on Figure 10. The way it could reduce the sensitivity to the varying number of links, is that it does not set a universal number of retrieved object to each query, rather it adjust the length of the list, based on how many of the objects are above the similarity threshold.

The method is applicable for any similarity measure, but bounded ones - such as cosine similarity - are better matches as their a thresholding comes more naturally.

The method was applied on the example data set, with the following results:

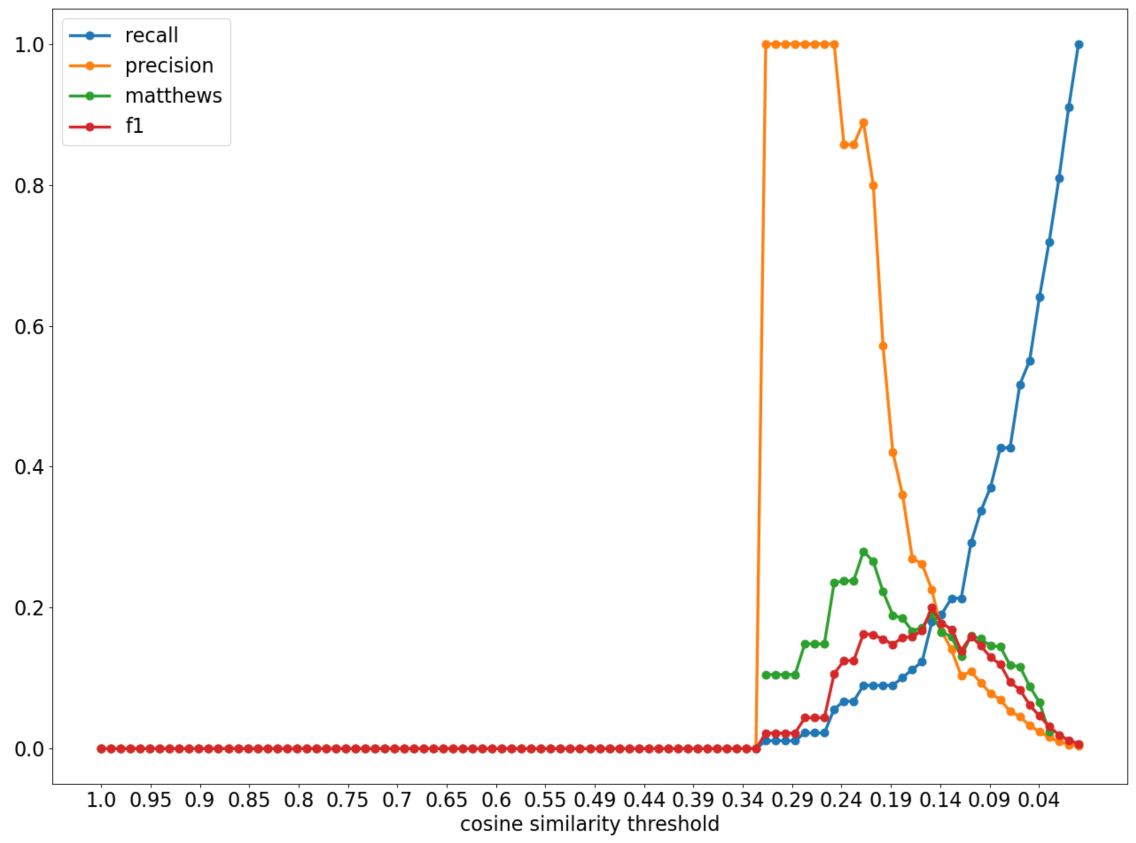


Figure 28: Recall, Precision, F1 and MCC on 80-20 division training set over similarity the similarity threshold

As it is visible, until a certain threshold (around 0.55) basically no object was retrieved. As the threshold was lowered the recall monotonously increased, while the precision peaked right after the first successful retrieval, remain maximal for an interval and then started to decrease. At zero similarity threshold, all objects were always retrieved, meaning a perfect recall and a close to zero precision. The MCC peaked right before the precision started to decrease, while the F1 score reached its maximum around the region of the intersection of the precision and recall.

The interesting part is that both the F1 and MCC measures' maximum was lower than in case of a fixed length retrieval list. The most likely explanation is that the variance of the similarity of retrieved objects over different queries is so high, that a single threshold can result in multiple (not relevant) returned elements for one query and possibly zero retrievals in another. This yields in overall lower performance metrics, therefore in the current project this approach is inferior compared to the list with a fixed length n.

4.6.1 Improvement possibilities

The approach is possible to improve, by defining separate threshold for separate object type relations, somewhat mitigating the difference in average similarity this way. On the other hand this increases the number of hyperparameters in the system, which is more difficult to keep maintained and also can be more project specific.

A solution to handle this issue is to adaptively set the threshold based on the available data set.

4.7 Results

The following section recaps and summarizes the performance evaluation of the previously introduced models and names the best performing one.

4.7.1 Train validation split

One of the main research questions was to set up a sufficient train-validation split. The first attempt was rather unsuccessful, but the second one solved the problem properly.

Object-based approach The first trial was unsatisfactory, as the degree distribution in the train and validation set was significantly different, which easily results in different performance evaluation on the two sets. Another issue was that some objects could not claim a clear train or validation set membership, as their connected objects were mixed.

Link-based approach The second method provided substantially more similar degree distributions - although could be further improved by a non-random weighting - and it also overcome of the challenge of the nodes connected to both groups, as in this approach only the links were labeled.

4.7.2 Evaluation criteria

The second important research goal was to set up a reliable evaluation criteria which enables the comparison of different models. The task was solved with the usage of traditional machine learning evaluation measures, such as precision, recall and aggregated measures as F1 score and the Matthews Correlation Coefficient.

4.7.3 Information retrieval system

The core research question, the actual topic of the thesis was to create an information retrieval system for recommending the most likely relevant documents (requirements) to a query document (another requirement or test specification). The task was solved a tf-idf based system, with different approaches in order to determine the number of retrieved elements. The hyperparameter tuning was handled by grid search. In the following the results of the different approaches are summarized.

Tf-idf with top n elements retrieved This version uses tf-idf to represent the documents in the corpus and then during retrieval simply a fixed length list of elements are retrieved for each query. The length n was determined simply as a hyperparameter. After hyperparameter optimization the model's best performance was 0.34 MCC and F1 score. This was the best performing model.

Tf-idf with similarity threshold This version also uses tf-idf, but instead of a fixed list length, a similarity threshold was defined. This resulted in a 0.28 MCC and a 0.23 F1 score.

Computation UI based optimization of the tf-idf system The alternative optimization remained in the current project more of a future possibility as several strong limitations were identified (see more at the relevant section), therefore a clear performance metric is not assigned.

BM25 experiments Besides the tf-idf based system, another function, the BM25 was also tested briefly. The Python implementation used in the prototype was not as customizable as the tf-idf model, therefore it was inferior - 0.28 MCC and F1 score - compared to the tf-idf after hyperparameter tuning.

5 Discussion

5.1 Limitations

5.1.1 Limited model performance

The first and most obvious limitations of the current prototype is its performance measured by the classical evaluation criteria (precision, recall, F1 score, MCC). The recall curve grows relatively slowly, while the precision drops quickly, which results in limited F1 and MCC scores. Based on the results, the system is useful with the current performance level, but nevertheless its utility clearly grows with a better performing information retrieval model. There are several more complex models, however it is not guaranteed that a substantially better level is achievable, because the nature of the data set (generally short textual content) is also a limiting factor.

5.1.2 Limited optimization criteria

Another limitation is that currently the best working optimization criteria was solely based on machine learning-like evaluation metrics, such as the F1 score, which does not take into account the specificities of the context where the model is used. The cost of false positives and false negatives are not necessarily equal, and in order to model the affect of the different performances on the user experience and usability, a more complex, computation user interface design based approach, similar to the one briefly described in the experiments chapter, could be a good direction of improvement.

5.2 Future development possibilities

Considering the limitations of the prototype introduced in the thesis, the following future improvements are recommended with a decreasing importance.

5.2.1 Integration to the Autoport tool

The possible ways of integration are described at the system architecture section, the prototype either needs a full C++ re-implementation or the extension of the prototype with a connection to the tool's database and an API to reach the IR system. It is clear, that this part is the most crucial before any other performance improvements.

5.2.2 Full implementation of the computational UI design based approach

An alternative approach was briefly presented, which uses computational UI design methods to determine the length of the shortlist. The weaknesses and missing parts, namely that it assumes zero decision time for object selection and the lack of constants, were pointed out and are necessary to be taken care of in order to use the approach in practise.

5.2.3 Word embeddings

A straightforward way to increase the system's overall usability is to aim for a higher performance through a more complex information retrieval model. A prefect example for this is through replacing the current bag of word methods with word embeddings, described in detail in related work.

5.2.4 Weighting objects by a priori distribution of object types

When tracing to multiple object types is allowed - e.g. we want to trace an object to a software requirement and it can either be a customer requirement or a test specification - the similarity scores of the individual objects can be weighted be the a priori probability of their object type's linkage to the query object.

The distribution in the number of links to a specific object type, or in other words the distribution in the degree of vertices, in the project until the time of the quiery could serve as the a priori probability distribution of adding an n^{th} link to a specific object type's instance.

5.2.5 Better than random weighting for train–validation split

When performing the train-validation split, each link gets a weight. Currently these weights are completely randomly assigned and although the degree distributions are relatively similar in the train and validation split, there are some differences, especially in the proportion of nodes with higher degrees. This is a know phenomena due to the random weights and can be improved if the weigh assignment is not random, but it takes into account the degree of the node to which the edge belongs.

5.3 Other interesting experiment ideas

5.3.1 Simulation of model performance over growing corpus

As more objects are added to the project, the project's corpus grows, therefore might be interesting to show how well the model works in different stages of the project. In the beginning there are relatively few options to recommend from, but also the models knowledge-base, the corpus is smaller. Over time the corpus grows, so the model is familiar with more terms, but the number of options also increase.

5.3.2 In some cases the tool shall mark that no objects are sufficiently similar

Based on the similarity threshold based selection of the shortlist – even when using the other approach with top n elements – if all of the objects' similarity is below a certain - considerably low - threshold, it might be better not to recommend anything.

6 Conclusion

After discussing the results with the thesis advisor from the company side, it was concluded, that the prototype proved to be useful in several ways and it is worth to further develop system and especially to integrate it to the Autoport tool at some point. The main limitations and future development possibilities were discussed in the previous chapter, here the main goal is to show the ways, in which the system is directly usable.

6.1 Original goal - to speed up engineering work

During the development of a project, as new requirements and test specifications are added, requirement engineers spend notable engineering hours with setting up the linking between the different objects in the Tool. The original goal was to develop a prototype tool, which speeds up the process of requirement tracing, with offering a shortlist of most relevant objects. This goal was clearly achieved, therefore it is logical to say, that information retrieval based systems, as the current prototype's tf-idf implementation, are worth to apply to the field of requirements engineering and the technology choice, with all of its limitations, was sensible.

During the development of the prototype, further potential use cases came up, in the following some of these are presented.

6.2 Sanity check for newly added links

In top of the recommending functionality, at any point in the development process the prototype can serve as a sanity check over all freshly added links between objects. This way it can provide feedback to the engineers about their expert choices, and either fortify the truth value of their choice - if it is aligned with the information retrieval prediction - or mark the new connection for another expert check up - if the choice is hard to explain for an information retrieval point of view. When checking each other's work, either as a supervisor or just as a peer review, the highlighted links should be validated more closely.

6.3 Security check of an existing project

Similarly to the sanity check of newly added links, the prototype system can evaluate all links between objects in an existing project and flag/highlight those which seem to be unrealistic based on their textual similarity. This way project maintained in a different requirements engineering tool and imported into the prototype system, can be checked before milestones or customer meetings.

6.4 Train - validation division and evaluation system

The other two side goals were to set up a proper train validation split method as well as a working evaluation system. At the results section it was already discussed, here it is only shortly mentioned, that these goals were also met, thus enabling the further development, testing and comparing of the prototype or alternative systems.

References

[1] Risto Tiusanen, “Autoport Project,” 2019.

[2] R. Tiusanen, T. Malm, E. Heikkilä, and J. Sarsama, “Safety approaches for autonomous mobile machines in industrial environments,” in *Assuring Safe Autonomy* (M. Parsons and M. Nicholson, eds.), pp. 405–411, 2 2020.

[3] A. Rantala, “Inter-organizational collaboration in software product development,” tech. rep., 2020.

[4] J. Vago, O. Witasse, P. Baglioni, A. Haldemann, G. Gianfiglio, T. Blancquaert, D. McCoy, R. Groot, and E. Team, “Exomars: ESA’s next step in Mars exploration,” *ESA bulletin. Bulletin ASE. European Space Agency*, vol. 155, pp. 12–23, 4 2013.

[5] K. Fowler, “IEEE Instrumentation & Measurement Magazine,” tech. rep., 2004.

[6] CDC, “Requirement Tracebility,” tech. rep., 2006.

[7] K. Mohan, P. Xu, L. Cao, and B. Ramesh, “Improving change management in software development: Integrating traceability and software configuration management,” *Decision Support Systems*, vol. 45, pp. 922–936, 11 2008.

[8] P. J. Roache, “Verification and Validation in Computational Science and Engineering,” tech. rep., 1998.

[9] W. L. Oberkampf and C. J. Roy, *Verification and validation in scientific computing.* Cambridge University Press, 2010.

[10] IBM Corporation, "IBM Engineering Requirements Management DOORS® and DOORS® Web Access V9.7 documentation," 2019.

[11] The Digital Project Manager, "The Best Requirements Management Tools Of 2020," 2020.

[12] I. Visure Solutions, "Visure - Requirements Management ALM platform," 2020.

[13] Phaedrus Systems Ltd, "Get To Know Visure - The Complete and flexible Requirements Engineering solution," 2017.

[14] Siemens Industry Software GmbH, "Polarion® REQUIREMENTS™," 2020.

[15] Polarion Software, "LiveDoc enhancements," 2014.

[16] C. D. Manning, P. Raghavan, and H. Schütze, *An introduction to Information Retrieval.* USA: Cambridge University Press, 2008.

[17] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," *Found. Trends Inf. Retr.*, vol. 3, pp. 333–389, 4 2009.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," 2013.

[19] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data," 2017.

[20] J. Brendl, *word2doc.* PhD thesis, 2018.

[21] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets.* USA: Cambridge University Press, 2nd ed., 2014.

[22] A. Oulasvirta, P. O. Kristenssonm, X. Bi, and A. Howes, *Computational interaction.* Oxford, United Kingdom: Oxford University Press, 2018.

[23] A. Oulasvirta, N. R. Dayama, M. Shiripour, M. John, and A. Karrenbauer, "Combinatorial Optimization of Graphical User Interface Designs," *Proceedings of the IEEE*, vol. 108, no. 3, pp. 434–464, 2020.

[24] D. Ahlström, "Modeling and Improving Selection in Cascading Pull-Down Menus Using Fitts' Law, the Steering Law and Force Fields," tech. rep., 2005.

[25] I. Scott MacKenzie, “Fitt’s Law as a Research and Design Tool in Human-Computer Interaction,” *Human-Computer Interaction*, vol. 7, pp. 91–139, 1992.

[26] S. K. Card, W. K. English, and B. J. Burr, “Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys, for text selection on a CRT,” *Human-Computer Interaction*, pp. 386–392, 1978.

[27] I. S. Mackenzie and R. W. Soukoreff, “Card, English, and Burr (1978)-25 Years Later,” tech. rep.

[28] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace, 2009.

[29] T. Oliphant, *Guide to NumPy.* USA: Trelgol Publishing, 2006.

[30] Wes McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference* (St\’efan van der Walt and Jarrod Millman, ed.), pp. 51 – 56, 2010.

[31] J. Paulett and D. Aguilar, “jsonpickle Documentation,” 2018.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, B. Michel V.
}and Thirion, O. Grisel, M. Blondel, R. Prettenhofer P.
}and Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[33] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[34] Dorian Brown, “Rank-BM25: A two line search engine,” 2019.

[35] Qt for Python, “Qt for Python.”

[36] Google, “Training and Test Sets: Splitting Data,” 2018.

[37] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognition Letters*, vol. 27, pp. 861–874, 6 2006.

[38] Y. Sasaki, “The truth of the F-measure,” tech. rep., 2007.

[39] B. W. Matthews, “Comparison of the predicted and observed secondary structure of T4 phage lysozyme,” *Biochimica et Biophysica Acta (BBA)-Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

List of Figures

List of Tables