HTW Chur
Hochschule für Technik und Wirtschaft
University of Applied Sciences

Churer Schriften
zur Informationswissenschaft
Herausgegeben von
Wolfgang Semar

Arbeitsbereich
Informationswissenschaft

Schrift 98

Counteracting Concept Drift in Natural Language Classifiers

Proposal for an Automated Method

Kirsten Scherer Auberson

Chur 2018

Churer Schriften zur Informationswissenschaft

Herausgegeben von Wolfgang Semar

Schrift 98

Counteracting Concept Drift in Natural Langage Classifiers

Proposal for an Automated Method

Kirsten Scherer Auberson

Diese Publikation entstand im Rahmen einer Thesis zum Master of Science FHO in Business Administration, Major Information and Data Management.

Referent: Prof. Dr. Ingo Barkow

Korreferent: Prof. Dr. habil. Wolfgang Semar

Verlag: Arbeitsbereich Informationswissenschaft

ISSN: 1660-945X

Chur, Dezember 2018

Kurzfassung

Natural Language Classifier helfen Unternehmen zunehmend dabei die Flut von Textdaten zu überwinden. Aber diese Classifier, einmal trainiert, verlieren mit der Zeit ihre Nützlichkeit. Sie bleiben statisch, aber die zugrundeliegende Domäne der Textdaten verändert sich: Ihre Genauigkeit nimmt aufgrund eines Phänomens ab, das als *Konzeptdrift* bekannt ist. Die Frage ist ob Konzeptdrift durch die Ausgabe eines Classifiers zuverlässig erkannt werden kann, und falls ja: ist es möglich dem durch nachtrainieren des Classifiers entgegenzuwirken.

Es wird eine System-Implementierung mittels Proof-of-Concept vorgestellt, bei der das Konfidenzmass des Classifiers zur Erkennung von Konzeptdrift verwendet wird. Der Classifier wird dann iterativ neu trainiert, indem er Stichproben mit niedrigem Konfidenzmass auswählt, sie korrigiert und im Trainingsset der nächsten Iteration verwendet. Die Leistung des Classifiers wird über die Zeit gemessen, und die Leistung des Systems beobachtet. Basierend darauf werden schließlich Empfehlungen gegeben, die sich bei der Implementierung solcher Systeme als nützlich erweisen können.

Schlagwörter: Natural Language Classification, Konzeptdrift, Text Retrieval, Klassifikation, Natural Language Processing, Machine Learning

Abstract

Natural Language Classifiers increasingly help Enterprises overcome the deluge of textual data coming their way. But these classifiers, once trained, lose their usefulness over time, as they remain static but the textual data's underlying domain evolves: Their accuracy decreases in a phenomenon known as *concept drift*. Can this phenomenon be reliably detected in the classifier's output? Once detected, can it be corrected through re-training, and if so, how?

A proof-of-concept implementation of a system is presented, in which the classifier's confidence metrics are used to detect concept drift. The classifier is then re-trained iteratively, by selecting test set samples with low confidence value, correcting them, and using them in the next iteration's training set. The classifier's performance is measured over time, and the system's performance is observed. Finally, recommendations based on this implementation are made, which may prove useful in implementing such systems.

Keywords: natural language classification, concept drift, text retrieval, classification, natural language processing, machine learning

Preface and acknowledgements

This paper attempts to bridge a gap between the theoretical scientific work on methods and techniques how to detect and counteract concept drift and the feasibility of these methods in real-world applications and settings.

The master thesis was motivated by various discussions about classification systems and how to improve them with clients and consultants from the author's professional environment. The aim is to both design and build a proof-of-concept for a text classification system that is able to (semi-)automatically detect and counteract concept drift.

The high-level architecture developed in the course of this paper should provide a good starting point for a practitioner (such as an IT development team) with building an effective text classification application.

I would like to thank my family for their unwavering support, especially my husband who makes one mean gin tonic. I would especially like to thank my husband and my good friends who proof-read everything single line.

You're awesome.

Contents

List of figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
CRM	Customer Relationship Management
CSV	Comma-separated values (AND Categorization Status Value)
CV	Confidence value
DNF	Disjunctive Normal Form
DSR	Design Science Research
FN	False Negatives
FP	False Positives
FR	Functional requirements
GDPR	General Data Protection Regulation
HITL	Human-in-the-Loop
HiTLCPS	Human-in-the-Loop cyber-physical system
ID	Identifier
IBM NLC	IBM Watson Natural Language Classifier
KMS	Knowledge Management System
ML	Machine Learning
NFR	Non-functional requirements
NLC	Natural Language Classifier
NLP	Natural Language Processing
PoC	Proof-of-Concept
REST	Representational State Transfer
SME	Subject Matter Expert
SOAP	Simple Object Access Protocol
SVM	Support Vector Machine
TC	Text classification
TN	True Negatives
TP	True Positives
UI	User Interface
UX	User Experience

1 Introduction

Individuals, business, government agencies and other organizations are confronted each day with large amounts of textual data in various forms, such as documents, e-mails or instant messaging. This immense body of text is critical for working and living, containing potentially enormous value. In its original form though, it is, due to the volume, difficult to extract that value (Manning & Schütze 1999 p. XXIX). As the importance of all of this text is usually an unknown factor, the tendency is to store and save almost everything. How many E-Mail inboxes are clutter-free?

Imagine such a situation where a sheer unmanageable number of objects exist from which a selection must be made. The best approach would be to categorize these objects first, by trying to structure them according to certain criteria first (Henrich 2008 p. 218). This superordinate structure organizes the objects, making them more manageable and accessible. Categorizing texts as such is also called text classification.

But trying to classify this huge amount of text manually is an insurmountable task. Therefore, systems to automatically classify this data are needed. However, this requires the classification system to be able to decide for a text in which category - class respectively - it belongs. This requires the system to have knowledge about the categorization rules: Why does one document belong to the category "music", and another to the category "movie"?

As humans, we are inherently able to understand the meaning of written text, of natural language, and are even able to infer the context from it, even if it's only implied. In its original state, natural language represents unstructured data for any kind of system. Unstructured data must then be parsed, tagged and processed before it can be interpreted (Hurwitz, Kaufman, & Bowles 2015 p. 39). A classification system therefore needs to be able to process this unstructured data in order to be able to classify it.

This is where Natural Language Processing comes in - the task of NLP is by applying various machine learning methods to translate unstructured content into meaningful information (Hurwitz et al. 2015 p. 40) by detecting underlying patterns. As a technology, it is an enabler to understand the meaning of unstructured data. (ibid. p. 53). The subfield of NLP concerned with classification is called natural language classification. Such a classifier can be applied within the context of a classification system to help identify the category a particular text needs to be sorted into, by detecting the patterns that correspond to a specific category.

1.1 The problem space

Such a text classification system is usually trained before productive use and is then deployed. If the environment where the text to be classified are coming from is not a closed

corpus, but a "living" system such as a knowledge management system or a customer care system, there is a high probability that the content will evolve, change over time. Text without a previous mapping in the original classes will thus enter the system, which may result in a false classification. This phenomenon is known as concept drift, an unforeseen change in the underlying patterns.

As time passes, this results in increasingly less accurate classification. In a business context, this might lead to delayed processing of a customer inquiry, or an invoice not to be paid automatically. In Health Care, this could even happen at the cost of lives.

1.2 Objective of the master thesis

The aim of the master thesis is to both design and build a proof-of-concept for a text classification system that is able to (semi-)automatically detect and counteract concept drift. A human-in-the-loop, whose feedback to the system is relevant for the classification accuracy, is an integral part of this design.

This text classification system should be able to

- Reduce misclassifications by detecting concept drift
- Detect resulting missing classes
- Reinforce correct classification
- And consequently, re-train the classifier with the help of a human-in-the-loop

Furthermore, open questions and problems arising during the conception and implementation of such a system should be identified and pointed out. Last but not least, the intent is to create a catalogue of recommendations that would be relevant for the implementation of the proof of concept for a real-world scenario. Since many assumptions were made within the PoC as well as laboratory conditions implemented, such as the HITL actions, the aim is to be able to facilitate a real-world implementation, where they can be translated into business requirements where applicable.

1.3 Placement within Information Science

Historically, document classification originated from library science, or more broadly from the field of information science. Information Science is a field that primarily studies the analysis, collection, classification, manipulation, storage, retrieval and protection of information (Stock & Stock 2013 p. 3).

Text classification, as described here, is a task of information seeking (Russell & Norvig 2016 p. 860), which falls in the domain of content-based document management, also known as Information Retrieval (IR) (Sebastiani 2002 p. 1).

IR deals with the search for information and with the representation, storage, and organization of knowledge. In the case of this paper, this concerns the identification and representation of categories or classes, which model a specific domain of knowledge or interest. The categorization of texts into these specific classes, and how it is done, produces information that is problem-related and adapted to a given context (Strauch 2004 p. 107).

According to the short definition of information science at the beginning, this master thesis spans the entire range of information science: The analysis of given texts through NLP results in a model representation (the classifier model) of underlying applied categories. This model representation is then used to classify new unknown text into classes, thus making them retrievable. The identification of new classes (the concept drift) produces new, previously not identified information relevant to the context in which the classifier operates.

1.4 Research design and methodology

In order to achieve these objectives, an approach guided by Design Science Research (DSR) methodology was chosen.

Behavioral science research designs, which are usually classified into either a qualitative or a quantitative approach, are used to support the development and justification of "theories that explain or predict" system phenomena or behavior (Hevner, March, Park, & Ram 2004 p. 76). Their goal is to discover the cause-and-effect correlation of the investigated problem (Österle et al. 2010 p. 667), for already existing research object(s) (Frauchiger 2017 p. 108).

Design Science Research on the other hand, addresses an existing problem in reversed order: the object, or rather artifact, to be critically researched and evaluated, needs to be first constructed based on an identified business need (Hevner et al. 2004 p. 79). "Knowledge and understanding of a[...] problem and its solution are [thus] acquired in the building and application of an artifact" (Hevner et al. 2004 p. 82). Therefore, the goal of Design Science Research is to "seek[s] a solution to a real-world problem of interest to practice" (Kuechler & Vaishnavi 2008 p. 6), which has been identified above. The relevance of that problem has been determined through structured depth interviews with subject matter experts.

The research object therefore is the research product and vice versa. In order to be able to design, build and evaluate such a product adequately, "the problem [...], its causes and contexts" (van Aken, Chandrasekaran, & Halman 2016 p. 6) need to be analyzed, and the required knowledge base provided. Design Science Research thus incorporates aspects of behavioral science research. This also ensures the scientific rigor of the research, and

therefore the master thesis. DSR is based on a rigorous Research Framework (Hevner et al. 2004 p. 80, available as Figure 23 in the appendix), to ensure both relevance and scientific rigor in the development of solutions.

The development of the master thesis closely followed this research framework, which also provides an overview of possible influential factors and the necessary methodologies for building and evaluating the research product. In contrast to a literature review in a behavioral science research design, where literature on an already existing research object is being reviewed, a literature review in DSR needs to follow the components of the solution to be created as well as its underlying problem. It will provide the necessary background information on topics such as concept drift or classifier confidence where it is needed.

Structure and organization of the literature review of the master thesis therefore differ from the familiar behavioral science literature review.

1.5 Organization of this master thesis

The master thesis is organized according to the DSR Methodology Process Model proposed by Peffers, Tuunanen, Rothenberger, & Chatterjee (2007 p. 54), adapted to reflect this master thesis and its structure.

Hevner et al. (2004 p. 82) states, that "[t]he fundamental principle of design-science research from which […] [the] seven guidelines are derived is that knowledge and understanding of a design problem and its solution are acquired in the building and application of an artifact."

These seven guidelines describe how a DSR should be carried out, and acted as a guideline during the development of this paper. Based on Peffers et al. (2007 p. 49), a finalized research should have the following characteristics:

- The research culminates in an artifact which is a solution to a currently unsolved and important problem.
- The "utility, quality and efficacy" (Hevner et al. 2004 p. 85) of the artifact is rigorously evaluated, and the research represents a clear contribution in the area of the artifact.
- Rigorous methods are applied in both the development of the artifact and its evaluation (Hevner et al. 2004 p. 83).
- The artifact development considers existing conditions in the problem environment, and the research process and its outcomes are communicated appropriately to both the technical and management-oriented audiences.

This work hopefully reflects all of these characteristics.

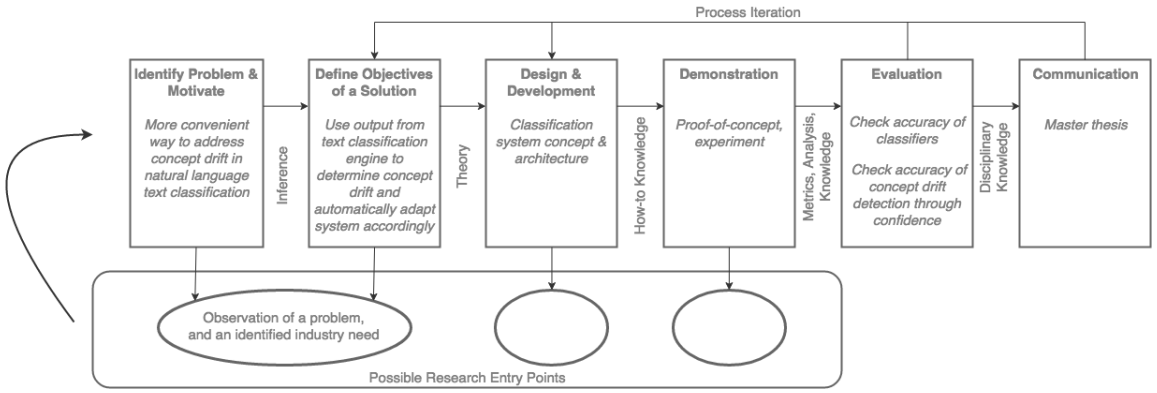

Figure 1: The DSRM Process Modell proposed by Peffers et al. (2007 p. 54) adapted for the master thesis

This paper is structured according to the DSRM process model depicted in Figure 1:

Chapter 2 describes the problem of concept drift in classification systems in more details and discusses the main problem components text classification and concept drift. Chapter 3 then defines the capabilities of a system to counteract concept drift and identifies the main solution components. The design and development of the architecture for this system is discussed in chapter 4, and the respective design artifacts are presented. Chapter 5 demonstrates the utility of the system through two experiments, which are evaluated in the following chapter. This paper concludes with open questions and recommendations for implementation in chapter 7, and chapter 8 summarizes the findings, and presents limitations to the master thesis as well as future research questions.

2 Identifying the problem: concept drift in text classification

Unstructured textual data such as mails, complaints and the like need to be sorted into folders or classes in order to make them retrievable. A classification, or categorization system allows the classification of such textual data. Due to the amount of unstructured text created each day, it is necessary in many cases that this classification is done automatically. A natural language classification system allows for automated classification of such data. This is becoming increasingly important due to "the increased availability of documents in digital form and the need to organize them" ((Liu, Loh, Youcef-Toumi, & Tor 2007 p. 171).

Text classification, usually done either based on rules or using a machine learning approach, is adding a level of retrievability to otherwise, due to the amount of data in question here, unretrievable information. When talking about text classification in this paper, it is usually in the context of NLP. NLP, short for Natural Language Processing, usually employs linguistic concepts such as part-of-speech and grammatical structure in order to "[...] extract a fuller meaning representation from free text. This can be put roughly as figuring out who did what to whom, when, where, how and why." (Kao & Poteet 2007 p. 1)

Such a text classification system is usually trained before productive use and is then deployed. As soon as such a system is in use, however, retraining of the system takes places only sporadically, and only if the client is willing to pay for it, usually when the number of mis-classifications are becoming troublesome. Often, if no action is triggered by an incoming text, no one even notices. The text classification system thus sometimes remains static for long periods. Misbeliefs about the capabilities of such a system, such as "It's an AI system, it will learn by itself"[1] also add to the problem

2.1 The problem with a static system

Natural Language text classification is increasingly adopted for the (pre)-processing of unstructured data coming in to companies through various channels and sources. This is part due to the constant expansion of available computing power, but also due to the steadily increasing number of unstructured texts produced each day.

When such large amounts of data have to be sorted, the classification and storage of this data is not trivial. A Natural Language Processing (NLP) classifier system can help classifying texts automatically and therefore ensure correct further processing. If we are not dealing with a closed corpus, but rather with a "living" system such as a knowledge management or customer care system, there is a high probability that the content will evolve

[1] This is a misconception about contemporary AI technology the author often encounters when talking to clients

and or change over time. Text without a previous mapping in the original classes will thus appear over time, resulting in a false classification.

Consequently, such a static system doesn't allow for:

- Recognition and / or correction of misclassifications, except individually and manually
- Reinforcement of correct classifications, or
- Training of new classes

Changes to the underlying class distribution of incoming text, called concept drift, can occur for different reasons, but are usually not accounted for in a set classification system. The resulting erroneous classifications may lead to a class system that is difficult to update. To account for concept drift, so far, according to Forster (in-depth interview, 17.05.2018) no convenient methods have been developed for practical application. Forster (2018) also states that various research papers are discussing the autonomous and automatic detection and handling approaches of concept drift as listed in Gama, Žliobaitė, Bifet, Pechenizkiy, & Bouchachia (2014). Forster (2018) noted though, that the existing measures discussed in these research papers would only work under lab conditions and specific experimental settings, but not in a real-world setting. They would therefore not be suitable to be used in an industry application (ibid.)

This statement is confirmed by Žliobaitė, Pechenizkiy, & Gama (2016 p. 109), who determine the research area of concept drift as being a quite young research field. Even though researchers realize the importance of concept drift for practical data mining applications, research problems have been addressed and formulated in artificial settings. They therefore conject that this research field will shift its focus from general detection and handling methods to more specific application-oriented approaches (Žliobaitė et al. 2016 p. 110).

Companies are therefore facing a lot of manual work as soon as concept drift is detected, if it is indeed detected at all. Forster (2018) states that companies usually respond to concept drift by first manually re-classifying the falsely categorized text or data, and then starting a new classifier training depending whether a concept drift is being perceived. This is a cumbersome process. A more practical approach would therefore be needed. The author received similar feedback during various discussions with customers, which frequently lamented about the problems with aging classifiers.

A solution that would enable companies respectively the users of such a system to easily detect concept drift would help with adjusting the classifier to the concept drift without adding too much additional costs in time or compute power. This identified "need" for a more convenient way to address concept drift thus triggered the idea for the development of this master thesis.

2.2 Components of the problem

The addressed problem space of concept drift within text classification consists of two components, each of which should be better understood before moving on to a possible solution. From the problem description, the following problem components can be identified:

Classification of text. Within this component, the following aspects should be examined:

- Definition and application: what is it and why is it relevant?
- The main approaches to text classification: Rule-based vs. machine learning
- Specifics and requirements of natural language text
- Single-class vs multi-class and single-label vs multi-label classification
- Natural language processing: what is it, and why is it of relevance in this topic?

Concept drift. For this component, the following aspects should be examined

- Definition and occurrence: what is it, and why does it happen?
- How can Concept Drift occur: Types of concept drift
- Strategies for handling concept drift - current solutions to detecting of and adapting to concept drift

2.2.1 Classification

Classification can be described as the task "[...]to classify a given data instance into a prespecified set of categories." (Feldman & Sanger 2007 p. 64) Imagine a situation where a sheer unmanageable number of objects exist from which a selection must be made. The best approach would be to classify the objects first, thus trying to structure the set of objects according to certain criteria first (Henrich 2008 p. 218), which are represented by their identified characteristics (Henrich 2008 p. 219). Classification therefore serves to increase the manageability of previously unmanageable objects.

Text classification

In our text classification problem (TC), also known as text categorization in the domain of document management, according to Feldman & Sanger (2007 p. 64), this would correlate to "[...]classify[ing] the topic or theme of a document." (Manning & Schütze 1999 p. 575) Or, as Sebastiani (2002 p. 1) puts it: "[...]labelling natural language texts with thematic categories from a predefined set."

Historically, document classification originated from library science, or more broadly from the field of information science. For the range of books existing in the present shelves, an

organisation system of some kind was needed to ensure that thematically similar books were placed close together (Henrich 2008 p. 217), making them more manageable. "Ever since writing was invented", Ferreira (2017 p. 2) states, "text was used to communicate crossing boundaries of time and space." Because of this, document or text classification, "[...]has always played an important part in organizing human life and society." (ibid.)

Current literature distinguishes two types of classification: rule-based, also referenced as knowledge engineering, and machine learning approaches (Feldman & Sanger 2007; Manning, Ragahvan, & Schütze 2009; Sebastiani 2002) We will look into these two approaches later.

The centuries old (third) approach of manual classification is neglected in the knowledge base used for this paper, because, as Manning et al. (2009 p. 255) phrase it: "A computer is not essential for classification. Many classification tasks have traditionally been solved manually. Books in a library are assigned Library of Congress categories by a librarian. But manual classification is expensive to scale."

This is especially true when moving away from the traditional playground of document - or text - classification into other contexts such as mail filtering (e.g. spam detection), genre classification (horror vs romance) or news stream filtering for relevant news, where a financial journalist for example only wants to read those news that have been labelled "mergers and acquisitions". (Manning & Schütze 1999 p. 575)

Massive amounts of information

Because of the technology available now, in particular mobile computing which allows people to basically write almost everywhere, massive amounts of data are created each day (Ferreira 2017 p. 2). In 2017, 16 million text messages, 156 million emails and 103,447,520 spam emails were sent every minute (Marr 2018). This puts into perspective the amount of text with which individuals, businesses and government agencies are confronted each day, "[...]that are critical for working and living, but not well enough understood to get the enormous value out of them that they potentially hide." (Manning & Schütze 1999 p. XXIX) As the importance of all of this text usually is an unknown factor, the tendency usually is to store and save almost everything. How many E-Mail inboxes are clutter-free?

This amount of text of "[...]highly variable length, content, and quality" (Jurafsky & Martin 2008 p. 5) created and stored everyday would make manual classification a lifetime task. Therefore, systems to automatically classify this data are needed.

A formalized representation

Text classification is a task in which a system has to decide, for a text of some kind, which of a predefined set of classes the text belongs to (Russell & Norvig 2016 p. 865). In an approach from a set theoretical point of view, this can also be formulated as follows:

For the text classification, we are given document d $\in$ D, where D is the document domain or corpus. A fixed set of manually pre-defined classes C = {c1, c2, . . ., c|C|} is available. Classes are also called categories or labels (Manning et al. 2009; Sebastiani 2002)

According to Sebastiani (2002 p. 3) text classification is simply the task of assigning a Boolean value TRUE (T) or FALSE (F) to each pair

⟨dj, ci⟩ $\in$ D, thus declaring the class ci for document dj as either true or false.

For D = {d1, d2} and C = {a, b}, a value T or F would have to be assigned to these four pairs.

Class \ Document	1	2
A	⟨d_1, a⟩	⟨d_2, a⟩
B	⟨d_1, b⟩	⟨d_2, b⟩

Figure 2: Document / Class pair table for value assignment (own depiction)

A value of T for the pair ⟨d1, a⟩ for example indicates the categorization of document *d1* under class *a*, while a value of F for the pair ⟨d2, a⟩ results in document *d2* not being categorized into class *a* (Sebastiani 2002 p. 3). See Figure 1 for a visualization of the document/class pairs. The task therefore is to approximate a yet unknown function F : D$\times$ C $\rightarrow$ {T, F} that describes how documents should be assigned to classes (Sebastiani 2002 p. 3; Feldman & Sanger 2007 p. 66). Manning et al. (2009 p. 256) reformulated that into γ : D $\rightarrow$ C, a "classification function γ that maps documents to classes".

Two classification approaches

This representation of the classification function allows us to assume any kind of underlying classification method. As stated before, there are two main approaches to text classification: The first is the knowledge engineering approach, also known as rule-based, where the knowledge of an expert about the classes "[...]is directly encoded into the system" (Feldman & Sanger 2007 p. 64) in the form of one logical classification rule per class or category (Sebastiani 2002 p. 8). Machine learning text classification on the other hand depends on an algorithm to automatically learn the classification rules, or as Manning et al. (2009 p. 255) put it "[...] the decision criterion of the text classifier" in a supervised learning process from training data.

Rule-based TC, also known as Knowledge Engineering

The rule-based approach to text classification depends on determining the word patterns that are the best representation of the different classes (Aggarwal & Zhai 2012 p. 165). These patterns are usually determined manually by a domain expert, and then - again manually - translated into a set of logical rules, where one rule defines the conditions for a document to be labelled with a given class (Sebastiani 2002 p. 8). These classification rules, used for the creation of subsequently automatic document classifiers, all have the same structure: The left-hand side approximates the word pattern detected in a class, and the right-hand side defines the class (Aggarwal & Zhai 2012 p. 178). They are generally expressed in Disjunctive Normal Form (DNF), which is a standardized representation of a function in Boolean logic. See Figure 3 for an example to classify a document into the WHEAT category

if ((*wheat & farm*) **or**
(*wheat & commodity*) **or**
(*bushels & export*) **or**
(*wheat & tonnes*) **or**
(*wheat & winter & NOT soft*)) **then** WHEAT **else** NOT WHEAT

Figure 3: Rule-based classifier for (a) Wheat class; keywords are indicated in italic, classes are indicated in SMALL CAPS (adapted from Sebastiani 2002 p. 9)

Another way to describe rule-based classification is to see it as a set of *If-this-then-that*. In this case, it means that a given document (a text) is categorized under the class WHEAT if it satisfies the above formula, in this specific case as long as it satisfies at least one of the clauses (Sebastiani 2002 p. 8).

This is of course a simplified representation of such a rule, but rule-based classification can be generalized as follows: The condition (*If this*) represents a combination of keywords that must all be present in a given document for the conclusion - the class - (*then that*) to be the consequence (Aggarwal & Zhai 2012 p. 179).

This will work fine *IF* all the relevant patterns under which a categorization decision can be made are known and encoded into rules. In order to anticipate the closer look at the machine learning approach, such a given, complete set of rules "[...] is essentially the model which is generated from the training data." (Aggarwal & Zhai 2012 p. 178) This corresponds to the way a machine learning classifier is built.

According to Manning et al. (2009 p. 255), rules coded in such a way are scalable, but creating and maintaining can be very labor intensive. A subject matter expert (SME) in the knowledge domain of the document space with good skills in writing regular expressions should be able to create rule sets that could "[...] rival or exceed the accuracy of [...]automatically generated classifiers." (ibid.) Manning et al. (2009 p. 335) estimate the time

required to create such a well-tuned rule at 2 days per class. This could prove to be a bottleneck for an organization though. To find someone with such a specialized skillset for such an extended amount of time could be difficult. Maintaining the classifier such as updating the classes due to e.g. concept drift, which will be discussed later on in this chapter, is not included in this time estimate. An even though such a classifier is supposed to be scalable, it cannot be ported to another knowledge domain without building the rules again from scratch (Sebastiani 2002 p. 9).

The machine learning approach

As with the rule-based approach where a subject matter expert identifies word patterns, many ML classifiers distinguish classes through word patterns (Manning et al. 2009 p. 289). Stanton (2013 p. 173) simplifies it such that ML basically is a process of identifying patterns in incoming information that correspond to a specific result. Through an inductive process a Machine Learner infers a generalization (the class allocation in the classifier) from available evidence in a set of pre-labeled data, also known as the training set. Evidence for a class would be the characteristics of the class, which again are the underlying patterns and word distributions.

The ability of a classifier algorithm to then deliver good classification performance on new input is a central challenge in machine learning according to Goodfellow, Bengio, & Courville (2016 p. 108), and is called generalization.

In the ML approach, this set of rules, which is laboriously built from scratch in the rule-based approach, is automatically learned during training from the carefully prepared training set. This is called supervised learning, because the learning process is guided, "supervised" by "[...] the knowledge of the categories and of the training instances that belong to them." (Sebastiani 2002 p. 9) According to Goodfellow et al. (2016 p. 103) supervised learning algorithms learn during the training process how to associate input (data to be classified) with specific output (class labels provided), given that the training data contains samples of possible inputs d for all possible outputs c. This means that the documents in the training set can be represented as $\langle dj, ci \rangle \in D$, all with an assigned value of TRUE.

The training set should be manually classified beforehand by an SME (Feldman & Sanger 2007 p. 64), which is much less time consuming than the rule definition process. According to Sebastiani (2002 p. 10), it is also easier, because it is fairly easy to characterize a concept (represented by a class) by selecting relevant instances of text. To describe a concept accurately in words, or even through a procedure on how to recognize an instance, is far more complicated.

Finding patterns: An excursion into Natural Language Processing

A classification system, like any system out there, be it a "[...] structured database, a query engine or a knowledge base, requires techniques and tools that enable the user to interpret the data (Hurwitz et al. 2015 p. 40). Unlike a structured database for example though, where schemas are applied to add meaning and understanding, unstructured data - as is natural language text - must be parsed, tagged and processed before it can be interpreted (Hurwitz et al. 2015 p. 39). It's important to decide which information to keep, and to find the patterns in that information that contain and produce "[...]meaning and context." (Hurwitz et al. 2015 p. 40)

This is where Natural Language Processing comes in - the task of NLP is to translate unstructured content into meaningful information (Hurwitz et al. 2015 p. 40). As a technology, it is an enabler to understand the meaning of unstructured data (Hurwitz et al. 2015 p. 53).

Therefore, NLP is, as Kao & Poteet (2007 p. 1) put it, the "[...] attempt to extract a fuller meaning representation from free text [...]" by applying a specific set of techniques (Hurwitz et al. 2015 p. 40) such as part-of-speech tagging or grammatical structure identification (Kao & Poteet 2007 p. 1).

This is done in order to "[...] determine something of the structure of text - normally at least enough that it can answer “Who did what to whom?”" (Manning & Schütze 1999 p. 17). An NLP system must therefore be able to make "[...] disambiguation decisions of word sense, word category, syntactic structure, and semantic scope." (Manning & Schütze 1999 p. 17)

A word by itself lacks context and content, much as telling somebody "28" without adding the measurement type (temperature) and the specifics (such as "outside, today"). Hence the task of an NLP system is to "[...] build [...] layers of contextual understanding by first looking to the left and right of that word to identify verb phrases, nouns, and other parts of speech." (Hurwitz et al. 2015 p. 42)

Natural Language Processing therefore serves to bring structure and understanding to data previously incomprehensible to machines, making it available for pattern detection.

Back to Machine Learning classification

The pre-labelled documents used for the training set are therefore the key resource in the ML approach (Sebastiani 2002 p. 10). In the best case they are readily available, because the organization that is implementing an automated text classification is switching from a manual classification to an automated process.

Labelled documents may also come from an implicit categorization, where an employee for example goes through all news articles in his feed reader in the morning and moves relevant articles in a specific folder like WHEAT (Manning et al. 2009 p. 255).

Because it is unstructured data, a text document must be processed accordingly before it can be used for training. This is similar to the rule-based approach, where the relevant keywords must be manually mapped to the class value. In the ML approach this is called feature selection, a process where the - for a specific class most relevant - features occurring in the training set are determined (Aggarwal & Zhai 2012 p. 166). In text classification, these are usually characters, terms, words and concepts (Feldman & Sanger 2007 p. 5) - as are in rule-based TC.

According to Bakis et al. (2017 p. 1), traditionally NLP techniques relied on manually designed features, which is called feature engineering, in order to build and enable the contextual understanding needed for NLP. This can be a very time-consuming task. Recent development in deep learning algorithms showed that they can learn multiple levels of features more or less automatically from the input data set and are very capable in "[...] tasks such as language modeling, named entity recognition, and sentiment analysis, without the need for manual feature engineering." (ibid.)

Through this, the documents are changed into manageable format. This also eliminates terms (features) that could increase the classification error on new data, also called noise features (Manning et al. 2009 p. 271).

The classifier algorithm then understands the pre-classified document ⟨dj, ci⟩ as set of feature/value pairs (Russell & Norvig 2016 p. 866), and can be trained on them. Training a classifier is also called "[...] constructing a model that captures the regularities in the examples of each category." (Ferreira 2017 p. 2)

In order for the classifier algorithm to perform well, a specific number of pre-classified documents is necessary. How many depends greatly on the type of deployed algorithm though. According to Manning et al. (2009 p. 335), some algorithms need hundreds or thousands of samples for each class in order to train a high-performance classifier, and many contexts contain large sets of classes. It is therefore advisable to compute a learning curve beforehand to determine the appropriate training set size (Manning & Schütze 1999 p. 587). Many training procedures are "computationally expensive" (ibid.) as well as time-consuming, which is why overly large training sets should be avoided (ibid.). A too small training set will result in lower classification accuracy though. The learning curve enables an informed decision how much documents are required for the desired performance (Manning & Schütze 1999 p. 587). The learning curve also provides a possibility to evaluate the probability of a classification error on new input. This is called the generalization error,

defined as "[…] the expected value of error on new input." (Goodfellow et al. 2016 p. 108) It is another point of view for measuring the performance of a classifier. Oversimplified, it could be defined as 1 minus accuracy.

Once the classifier is trained, the regularities can then be identified in new, yet unseen documents, and the classifier model can automatically assign a class (Ferreira 2017 p. 6).

Before such a classifier is used for classification within a business application though, it is good practice to see how it is performing on a test set (Manning & Schütze 1999 p. 577) The test set should contain no documents previously used in the training set. A classifier usually performs well on the data it was trained on, which is why the classifier evaluation should be done, according to Manning & Schütze (1999 p. 577), "[...] on a representative sample of unseen data since that is the only measure that will tell us about actual performance in an application." Evaluation is usually done using a confusion matrix, from which appropriate metrics for classifier performance are computed.

How many classes for a document?

The representation of the assignment function depicted in Figure 2 assigns exactly one class to a document. This is a so-called “one-of classification” (or “single-label classification”), where the classes are mutually exclusive (Manning et al. 2009 p. 306). If more than two classes are involved, this is called multinomial or multiclass TC. Classifying samples into only two classes (TRUE and FALSE) is a special case of single-label classification and called binary classification. Feldman & Sanger (2007 p. 67) state that it is also the most important classification case, "[...] because it is the simplest, most common, and most often used for the demonstration of categorization techniques."

If a document can belong any number of class (1-i) though, a classifier is confronted with the so-called any-of problem (Manning et al. 2009 p. 257), which is a more common classification problem than one-of (Manning et al. 2009 p. 306). This type of TC is called multi-label.

For this master thesis only one-of classification is considered, where a document belongs to only one class. Due to the scope and goal of this master thesis, methods and techniques of machine learning classification will not be further explored. A comprehensive view on these topics can be found in Aggarwal & Zhai 2012, Manning et al. 2009, Manning & Schütze 1999 and Sebastiani 2002.

Why ML?

The missing portability of a rule-based classifier as well as its high cost in terms of expert manpower are the reasons for increased interest in and application of machine learning TC (Sebastiani 2002 p. 1). In addition to that, the performance and effectiveness of machine

learning classifiers are steadily growing, with "[...] effectiveness levels comparable to those of trained professionals" ((Sebastiani 2002 p. 47), with the expectation that it will outperform rule-based classification eventually (ibid.).

With machine learning classification, the manual task of building rules and adapting them each time something changes, is removed. The application of ML to TC makes classification more convenient and cost-effective and adds a previously unknown portability to a classification system.

As touched on before, due to the increasing number of text documents created every day, the short response time required by most current applications for text classification, such as news filtering or personalization of social media content, make the manual approach unfeasible (Sebastiani 2002 p. 47, Ferreira 2017 p. 2). Automatic classification is needed to accomplish this.

2.2.2 Concept Drift

As seen before, a classifier needs to be trained before being able to classify new documents in an application. The necessary training set usually comes from historical, stationary data, which results in a model representing the concepts of interest as they were at the time when the training data was collected and labelled (Widmer & Kubat 1996 p. 965).

A concept, generally speaking, can be defined as "an abstract or generic idea generalized from particular instances."[2] In a classifier, a concept is represented as a class assignment. According to Webb, Hyde, Cao, Nguyen, & Petitjean (2016 p. 968) a concept can be defined as "[...] a set of vectors of X values such that any object with any vector of values in the set belongs to the concept, or equivalently, [a] [...] function X-> Y." This means that a concept is modelled based on the features that best represent the desired class and determine the regularities in the class samples (Ferreira 2017 p. 2). Again - it is the inductive process of generalizing the concept (the class) from the specific instances of the class representation in the training set. Comparing the concept function with the classification definition and function on page 11 in the section "A formalized representation" we can see that concepts and classes are the same, but concepts serve as a generalized term for the underlying representation models. A class could also be referred to as e.g. a category or a pattern, depending on the context.

The assumption is that the training data and the new data to be classified are similar (Manning et al. 2009 p. 258) and do not change over time. The training data is supposed to represent the real-world data in the future (Hand 2006 p. 7). This also means that they come

[2] Definition from Merriam Webster. https://www.merriam-webster.com/dictionary/concept [28/07/2018]

from the same source (Žliobaitė et al. 2016 p. 92) such as a specific population (Hand 2006 p. 7) with a similar underlying distribution of features. But, as Manning et al. (2009 p. 257) put it, "[…] high accuracy on the training set in general does not mean that the classifier will work well on new data in an application."

Because the environment where these yet unseen, to be classified documents are coming from is not stationary, it's dynamic. The underlying distribution in the real-world application data may therefore be nonstationary, since the source and the environment where the data is generated, might change over time (Sun, Tang, Zhu, & Yao 2018 p. 1). Hence the way concepts are represented in the real world may change over time. These unexpected changes in the underlying data distribution over time is referred to as concept drift. It is a "[...] generic term to describe computational problems with changes over time" (Žliobaitė et al. 2016 p. 93) in machine learning, predictive analytics and data mining. This results in classification from models trained with historical data to become less and less accurate over time (Žliobaitė et al. 2016 p. 92).

According to Gama et al. (2014 p. 2), real concept drift refers to a change in the conditions for a target class on a given input (identified features), while the distribution of the input could stay constant. Real concept drift in this case just refers to one specific type of concept drift.

Real concept drift could e.g. occur due to a change in personal interest for a user following an online news stream. The distribution of the incoming news might not change, but the condition for a news document to be classified as interesting or not for that specific user did (Gama et al. 2014 p. 2).

Other reason for changes in the underlying distributions are population changes (e.g. in a crisis, salaries tend to be lower), adversary activities (e.g. novel actions are employed in order to commit credit card frauds) or could be due to a complex nature of the environment (in automated vehicle navigation, for example, the landscape is very complex) (Žliobaitė et al. 2016 p. 92 and 97).

Another cause for decreasing classifier accuracy or increasing classification error respectively is of course noise. Noise results from a misleading feature or document that misleads the learner when included in the training set, thus increasing classification error (Manning et al. 2009 p. 303). According to Widmer & Kubat (1996 p. 21) as well as Tsymbal (2004 p. 1), it is very difficult to distinguish between true concept drift and irregularities because of noise in the training data. More information on noise and how to handle it can be found in Brodley & Friedl (1999).

Types of concept drift

Gama et al. (2014 p. 4)[3] and Widmer & Kubat (1993 p. 228) both distinguish two different types of concept drift.

Real concept drift that refers to changes in the world, is reflected in the classification results as a change in the data distribution where document d is assigned another class. This can happen without a change in the distribution of D (p. 4).

An example for real concept drift could be in fashion, for example clothing or music (Widmer & Kubat 1993 p. 228). The music itself doesn't really change, but what changes is the perception of what would be modern. Or when a new music genre is coming into being, like e.g "Industrial metal". The Band "Einstürzende Neubauten" existed before that music genre but could be categorized as Industrial music only after the Industrial genre developed in the late 1980s. Webb et al. (2016 p. 974) call this as a "novel class appearance", which is "[...] a special case of concept drift in which a new class comes into existence."

Virtual concept drift on the other hand refers to a change in the distribution of the incoming data without affecting the class assignments (p. 4). Here the opinions of Gama et al. (2014) and Widmer & Kubat (1993 p. 228) differ. According to Widmer & Kubat (ibid.), virtual drift doesn't really occur, but can be attributed to an incomplete data representation in the training data (p4). Virtual drift could be exemplified by the following example (taken from Gama et al. 2014 p. 5): Consider an online news stream of articles on real estate. The task for a given user is to classify the incoming news into relevant and not relevant. Suppose that the user is searching for a new apartment to buy; news on residential buildings and apartments is relevant, but holiday homes are not relevant. If the editor of the news portal changes, the writing style changes as well, but the residential buildings remain relevant for the user.

Figure 4 taken from Gama et al. (2014 p. 5) illustrates these two types of drift. The plots show that real concept drift changes the distribution of the documents, whereas in virtual drift only the distribution of the data changes, but not their assigned classes (Here described as the probability distribution of classes $p(y|X)$, where y = classes and X = input variables or feature set). Both types of drift may appear in combination. Typically, methods for handling real concept drift rely on feedback about the performance of the classifier, i.e. how good the accuracy is (p. 5).

[3] Unless stated otherwise, this section references the work of Gama et al (2014), with pointers to the relevant pages.

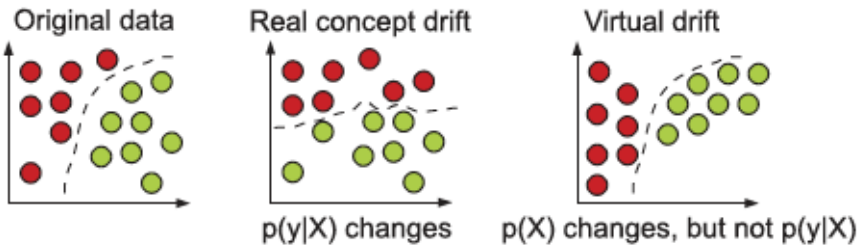

Figure 4: Types of drifts: circles represent samples; different colors represent the different classes (Gama et al. 2014 p. 5)

Change patterns can categorized by transition speed (sudden vs. incremental) and occurrence (sudden vs gradual). These patterns are illustrated in Figure 5 taken from Gama et al. (2014 p. 6).

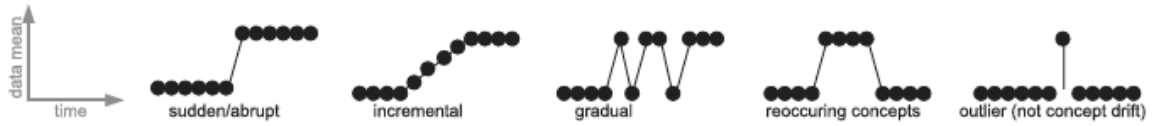

Figure 5: Patterns of changes over time, outlier is not concept drift (Gama et al. 2014 p. 6)

A sudden drift, changing from one concept to another, may happen because a sensor in a car is being replaced with another sensor that has a different calibration. An incremental change could happen because the sensor deteriorates and becomes less accurate, resulting in many intermediate concepts in between. A gradual change can occur e.g. in interest in news topics, when the user becomes more and more interested in holiday homes but won't switch suddenly, but instead goes back to his other topic of interest for some time. (Examples taken from p. 6).

As mentioned before, handling noise correctly and not confusing it with true drift is important. A drift may also recur from time to time, for example in fashion (p. 6). A more in-depth exploration of concept drift can be found in Webb et al. (2016), with an extensive taxonomy of drift categories (Webb et al. 2016 p. 922 onwards).

Addressing concept drift

Concept drift can occur in all non-stationary dynamic environments which continuously collect and categorize data over extended amounts of time. Predictive models (models that predict the category of data, as they can never be 100% sure) that are applied in such settings need to have mechanisms available to detect and adapt to concept drift. Classification accuracy will otherwise degrade. (Gama et al. 2014 p. 6, Webb et al. 2016 p. 965, Žliobaitė et al. 2016 p. 92)

As there are various types of concept drift which can occur in various settings, a "one-size-fits-all" solution is hardly feasible (Žliobaitė et al. 2016 p. 93). But rather, different concept drift detection and handling schemes may be required for each situation. To explore all the possible concept handling approaches would go well beyond the scope of this work.

Instead, we will just have a brief look at a few, namely

- Sliding window
- Uncertainty sampling
- Change detection

More information on how to approach and handle concept drift can be found in the works of Žliobaitė, Bifet, Pfahringer, & Holmes 2014 and Žliobaitė et al. 2016 and Gama et al. (2014).

Sliding window

The sliding window approach is a method where the classifier is updated (re-trained) on a regular basis. It is also called a blind adaptation method, as the model is adapted without an explicit need to (Gama et al. 2014 p. 21). As the concepts are known to change over time, a learning algorithm trusts only the latest samples available to be an accurate representation of the current state (Widmer & Kubat 1996 p. 2). The older the examples, the higher the possibility of an outdated context representation (Widmer & Kubat 1993 p. 229). The system keeps a set of current samples, both negative and positive. These represent the current environment of the system which should be correctly described by the current classification function, the description (ibid.). This is depicted in Figure 6 (Widmer & Kubat 1996 p. 3):

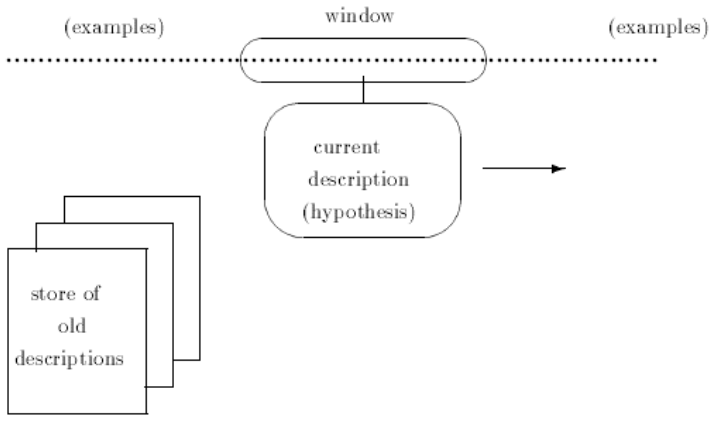

Figure 6: Current and old concept descriptions and the window moving along the stream of examples (Widmer & Kubat 1996 p. 3)

New samples are continuously added to the sample set (the window), while older examples are usually removed from it (ibid.). This is also called a forgetting strategy (Widmer & Kubat 1996 p. 3). According to Forman & George (2006 p. 225), the sliding window method is not

suitable for sudden or extreme concept drift. The system is unable to react flexibly due to the fixed window size (Widmer & Kubat 1993 p. 233).

A narrow window usually has too little samples to enable a good and stable accuracy but assures a fast adaptability in phases with concept change. A wide window on the other hand, due to the amount of representations of the old concept, will slow down the reaction to concept drift, but produces good results. An ideal setting therefore would be an adaptive one, where the window size could be adapted to the requirements.

Uncertainty sampling

Uncertainty sampling is a very simple and probably the most common strategy in active learning (Žliobaitė et al. 2014 p. 31). An active learning system can query the user or another information source for feedback, but only for a selected number of labels instead of all. These are usually those samples with the highest uncertainty, i.e. the classifier has a confidence about the probability of that specific label below a certain threshold. So, if the probability confidence for a class label for a given instance is lower than the threshold, the correct label is requested from the information source. Once the true label is assigned, the sample is used for learning. Žliobaitė et al. (2014 p. 31) call this the "Fixed Uncertainty Strategy". Of course, falsely classified samples with a higher confidence will be missed (Žliobaitė et al. 2014 p. 27). The confidence threshold for feedback query should therefore be regularly monitored, for examples in times with degrading overall confidence.

Change detection

Change detection refers to the mechanisms and techniques for explicit drift detection (Gama et al. 2014 p. 14). By indicating change points or small time-windows where change occurs, concept drift can be detected. Tsymbal (2004 pp. 5–6) identifies two indicators for detecting concept drift without the need for user feedback. The first is the average overall classifier confidence in correct class predictions on new samples. This can be achieved by regularly recording the confidence and computing the appropriate mean. This could simply be the arithmetic mean, as it would react immediately to low values. As all confidence values are always between 0 and 1, which will be discussed in more depth in the chapter "Classifier confidence" on page 36, there can be no outliers. The arithmetic mean is very sensitive to changes in the occurring values, which serves as a perfect indicator for a change in overall classifier confidence. Plotting these as a time series could provide an indicator for a concept change. The second indicator observes the number of samples below a specific confidence threshold. A rising number could indicate a possible concept drift.

These are the relevant approaches to concept drift for the presented master thesis. The next chapter will look at the classifier confidence in more detail.

3 Objectives of a solution: how can concept drift be counteracted?

As we have seen, concept drift, an unforeseen change in the properties of the target variable that the model tries to determine (Tsymbal 2004 p. 1), can prove to be an obstacle in a dynamic environment. With passing time, classification becomes increasingly less accurate.

In a business context, this might lead to delayed processing of a customer enquiry, or an invoice not to be paid automatically. In Healthcare, this could even happen at the cost of lives. For the sake of simplicity, we will assume a non-critical context for the scope of this paper and exclude industries such as Financial industries, Healthcare or Life Sciences for our assumptions. According to Forster (2018) mis-classifications would then require manual and individual intervention, which again is expensive and needs to be prevented if and where possible.

A text classification system adapted to business requirement should therefore be able to handle concept drift in the following ways (items 2-4 according to Tsymbal 2004 p. 2):

1. Detect concept drift
2. Quickly adapt to it
3. Distinguish noise from concept drift
4. Recognize and treat recurring contexts

Widmer & Kubat (1993 p. 228) state, that in order to detect concept drift, such a system would need "feedback from its classification attempts, to update the internal concept description whenever the prediction accuracy decreases" (ibid.). Such a feedback could be e.g. the classifier confidence.

This chapter introduces the conceptualization of such a system. The goal is not to develop an algorithm though, but to present an architecture for a system that will cover the items 1 and 2 from the previous list. This is done by applying the classifier confidence to detect concept drift, and human intervention, a "human-in-the-loop" (HITL) to

- Validate the detected concept drift
- Sift through noise, i.e. reinforce the classification
- Apply new classes where necessary

The aim of the solution presented in this paper, can therefore be formulated as follows:

The objective of the solution conceptualization is to provide a way to implement a system that is, with the human-in-the-loop, capable of semi-automatically adjusting to changes in the underlying class distribution of incoming text. It should be able to

- Reduce misclassifications by detecting concept drift
- Detect resulting missing classes
- Reinforce correct classification and
- Consequently, re-train the classifier

The underlying concept for such a system is presented in Figure 7. It is based on two assumptions:

I. Classifier confidence is sufficient to successfully detect concept drift in unstructured text
II. Implementing a system for the human-in-the-loop influences and enables the handling of concept drift in a classification system, thus supporting the practicability and applicability of such a system

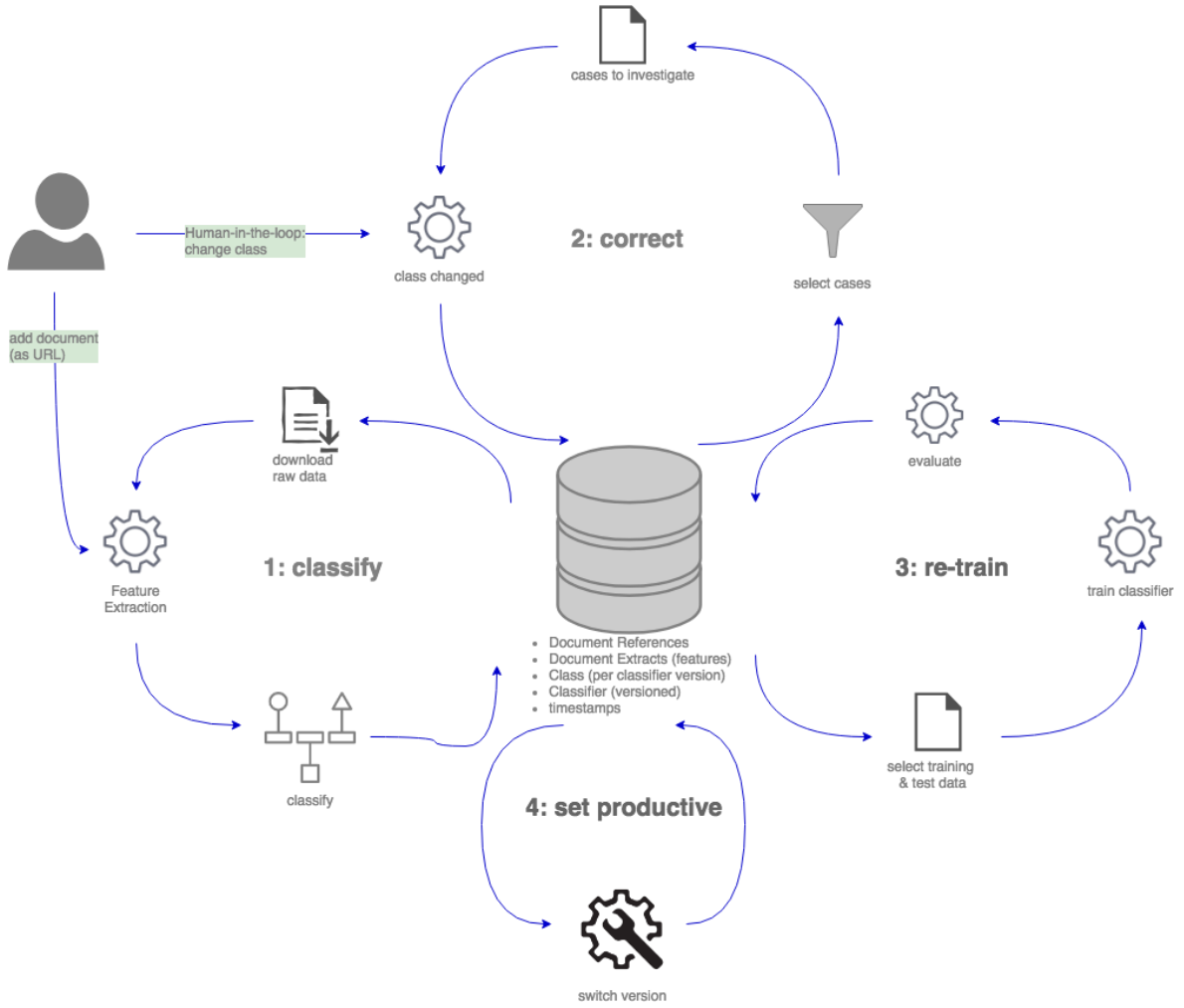

Figure 7: System design: Life cycle of a continuously trained classification system (proprietary depiction)

The system concept represents the life cycle of a continuously post-training classification system and thus forms the basis for the proof-of-concept (PoC) to be produced.

The aim behind this system is to automatically collect and reclassify data with a low confidence (e.g. < 80%[4]) based on the probability that a classifier has identified the correct class.

The life cycle consists of the following 4 steps:

1. Classify: A system, for example a business application, in which texts (e.g. documents in a collaboration space, or posts in a forum) are sorted into categories with the help of a classifier.
2. Correct: From the batch of freshly classified texts, those with a low classifier confidence are collected, and made accessible to the HITL. The lower the confidence, the less likely it is (so the assumption) that the correct class has been assigned to the

[4] The confidence threshold is based on the professional experience of J. Forster (2018)

document. The human-in-the-loop then confirms or rejects the assigned classes. In the latter case, either another, already existing class or a completely new class is assigned to the texts. The texts with their newly assigned or confirmed classes are then played back into the system.

3. Re-train: With a sufficient number of correct samples - confirmed as well as relabeled texts - a new classification model is created and evaluated
4. Set productive: the new classification model replaces the one used in production for Step 1 (classify). This is probably a manual step, but nevertheless an important one, where the performance of the model is evaluated and compared to that of the model currently used in production. If the employed algorithm(s) or classification allow so, the old model should be versioned.

Storage & Versioning: The data, such as label assignment, correction or timestamp, and models produced in all steps are stored and, if necessary, versioned. Storage & versioning is a central aspect and is therefore not presented as an individual step in the system concept.

The preceding step, the training and implementation of the classifier itself, is implied here. It can be seen as a trivial variant of steps 3 ("re-train") and 4 ("set productive"). We will go into more detail on training specifics in the section "Step 3: re-train"

Steps 2 and 3 are automatically executed at fixed intervals. These intervals are not time-based but rather controlled by the number of texts per new class, respectively the number of confirmed or re-labelled classes. The volume is determined by the number of documents required for the classification training, the sample threshold. The sample size is computed by training the classifier with a growing number of samples per class (a learning curve) - once the accuracy of the classification prediction has reached a satisfactory level, this will be the sample threshold needed to trigger step 3.

A learning curve has been computed for the IBM Watson Natural Language Classifier (NLC) and can be found in section "Step 3: re-train", Figure 8).

Assumptions

After this brief description of the system, the initial two assumptions can now be broken down into hypotheses. Because the system was conceptualized based on the two assumptions stated on page 28, the step of formulating the research questions was skipped here. A research question specifies what a researcher wants to find out through the research (Bryman & Bell 2015 p. 10), where a hypothesis represents an "informed speculation" about a condition or situation, set up to be tested (Bryman & Bell 2015 p. 92).

With a research question, we would want to find out whether concept drift could be detected by analyzing the variance in the classifier confidence. Since we already assume that this is true[5], the next step is now to test whether that actually *is* true.

> ***Hypothesis 1****: Concept drift in unstructured texts can be detected successfully through the analysis of variation over time in the classifier confidence. Classifier confidence therefore is a reliable detector for concept drift.*

One other aspect of the system design is the automatic semi-random compilation of training data for new classifiers. It is therefore necessary to look into the classifiers created in that way. The second hypothesis would read accordingly:

> ***Hypothesis 2:*** *The inclusion in the training set of automatically selected texts, that have been recently classified (labeled), then reviewed and corrected through HITL, results in a good classification model with an increase in accuracy compared to previous training instances.*

The HITL aspect is most probably represented by a number of people, all some kind of experts in their own fields, with specific responsibilities.

As the purpose of the system is to not only detect concept drift, but also to counteract it, the third hypothesis is as follows:

> ***Hypothesis 3:*** *Concept drift can be counteracted by applying the approach described in H2, where those recently classified texts that have been reviewed and corrected by HITL make up an integral part of the training set. A model trained in such a way is able to compensate for concept drift by adding the missing class(es) identified in the review process.*

3.1 System conceptualization: what happens in each step?

Each step has different tasks and requirements. Due to scope and scale of the master thesis, only steps 2 and 3 are further investigated in this paper. Steps 1 and 4 are nevertheless briefly described in this chapter, as well as included in the chapter "Designing the solution: a high-level architecture", where the underlying architectural considerations and decisions are presented and discussed. This section describes the four steps as well as the storage aspect in more detail, as well as their specific tasks and requirements.

[5] A definition of "assumption" according to Merriam Webster: "assuming that something is true". https://www.merriam-webster.com/dictionary/assumption [26/07/2018]

Storage and Versioning: a central aspect

Categorized data can trigger business-relevant actions directly without first having to be processed by a person (think credit card fraud detection resulting in a blocked credit card). Based on various discussions with clients and consultants from the professional environment of the author, it is assumed for the master thesis that the life cycle of such a classification system ends with Step 4.

To enable successive training though, it is necessary to continue to make labelled data available for processing, regardless of the business-relevant system into which it has been categorized. It is therefore necessary to add a data storage or an object store to the business application containing the classification system, enabling retrospective inspection / review of the classification results. This is represented by the "Storage and Versioning" aspect.

As previously described, the aim of the system proposed in this paper is to identify and correct misclassifications, with a focus on identifying new emerging classes that could not have been trained yet. This can counteract - or at least reduce - the risk of concept drift. This is done by manually correcting those labels assignment with a confidence value below a specific threshold, in our case 80%. This can either mean that the existing label of a text is confirmed, one of the other existing labels is assigned, or that a new, not yet trained on class is determined for this text (section "Step 2: correct").

For this purpose, the classification result must be saved, i.e. which label was assigned to a text together with the associated confidence value, as well as the classifier model (version) used for the classification. This classification metadata must be linked with a reference to the original text in order to be able to access it again if necessary.

This could be necessary, for example, to reflect a reclassification in the business application.

The previously created features that were used to classify the document are usually no longer required and can be deleted after the classification process has been completed. Handling of this temporary data can be done in two ways, depending on the specific system design.

I. Store it as classifier metadata. In the case of a new classifier training, the new training and test sets can be compiled without any further processing steps from the feature set corpus. This would also make sense in case of evolving documents, which could lead to noise in the training set if the features where extracted anew every time the document was used for classifier training.
II. Lose it. Instead, store an immutable document reference pointing to the original text, such as for example a Git Commit ID. The feature extraction process needs to be applied every time the text is used as a sample in a training or test set. In case of e.g.

a reclassification in the business application, this guarantees that the extracted features are always up to date and correctly reflecting their assigned labels. However, always provided that the feature extraction process is also up to date.

In our case, the feature set is stored in the object store. The reason for that is the fact that IBM NLC does not use many features, as it's currently only possible to train on 1024 characters (IBM n.d.-a p. 9). This represents only a modest amount of storage space, which is why it might be opportune to store the feature set for each text in the data storage. The trade-off here is storage space versus CPU and network load, but also execution time: It is much faster to process features stored locally rather than fetch data from the external system over the network.

We have seen that in order for the proposed system to work, different data is required and/or produced in each step. In the description of the individual steps, the data that needs to be stored will therefore be determined in each respective step.

Based on this, a fourth hypothesis concerning the data foundation for re-training decisions could be developed:

> ***Hypothesis 4:*** *In order for the concept drift handling classification system to work, and to be able to implement the approaches described in H1 and H2, it is necessary to store and process classification metadata separately.*

This is due to the fact that for the proposed system to work, not all data are available in the external system representing the application environment. The information about the classification, i.e. which samples have been reviewed and corrected, or the associated confidence for that label assignment, is necessary for the re-training process. This classification metadata can, of course, not exist in the External system and must therefore be stored and made available for processing

3.1.1 Step 1: classify

A classifier (more about classifier training in section 3.1.4) trained on pre-labelled historical data from the environment the classifier is applied to, is used to classify texts, e.g. in a live system. These texts represent raw data, on which sometimes a feature extraction process needs to be applied to. This is a necessary step to enable the classifier to process unstructured data (see chapter 2.2.1, section "Back to Machine Learning classification"). However, the feature extraction process and its implemented feature selection steps depends on the classifier used; in this system the classifier specified is the IBM Watson Natural Language Classifier, a cloud service. Since the IBM NLC is only able to classify texts with less than 1024 characters, data to be reclassified must be prepared accordingly. This includes specifically to remove possible noise features such as author or time stamps,

remove special characters, and reduce its length (if possible to a full sentence or word). This is a necessary step for all instances where the classifier is involved - initial training, classification, and re-training.

The labelled texts are then stored and or sorted into folders according to the class with the highest assigned classifier confidence.

Resulting data to be stored / saved:

- Feature set (the processed text)
- Classification data: label, confidence, timestamp, classifier (version)
- A document reference for future use (depending on the implemented classifier algorithm(s))

3.1.2 Step 2: correct

The highest classifier confidence value, which is ultimately responsible for the labeling decision, can still be below the confidence threshold. The lower the confidence value, the higher the classifier uncertainty. The underlying assumption here is that a classifier will assign labels with a much lower confidence, if incoming texts deviate from the classifiers current state of patterns, as could happen in concept drift. The lower the confidence, the higher the chance for misclassification. An uncertainty sampling method (section "Uncertainty sampling", page 22) is applied here.

Recently labelled texts with a confidence value below the threshold are collected and sent to a human agent (human-in-the-loop) with the text, the assigned class as well as the respective confidence. In this case, it might be sufficient to just send the output from the feature extraction process. With another set of algorithms, it might be necessary to send the raw, original text. The HITL then corrects the uncertain classification to the effect that either

- the originally assigned label is confirmed (a reinforced label)
- another class already existing in the label set is manually assigned
- a new class not yet existing in the label set is assigned. If it is the first instance of the new class, it is added to the list of temporary classes waiting to be trained. If the new class is recurring, which could indicate a concept drift, the count of available samples for re-training is increased by one.

The texts with their newly assigned or confirmed classes are then played back into the system. Label corrections or confirmations need to be stored together with the classification results. The information about the corrective HITL actions need to be stored in order to make these samples available for future re-training of the classifier. Texts confirmed or corrected

into an existing class serve to model the current environment of the classifier. A type of sliding window techniques is thus applied here.

Resulting data to be stored / saved:

- Corrected classification data: corrected label

3.1.3 Step 3: re-train

Once the sample threshold for a new class is reached, a new training set is compiled by

1. collecting all samples for the new class(es) that have reached the sample threshold,
2. collecting corrected samples for the existing classes,
3. adding random samples from existing classes in order to reach the desired training set size (if necessary).

The appropriate size of the training set depends on the selected method but can be determined by computing a learning curve.

Too large training sets should be avoided, as a training process can be computationally expensive and could take a lot of time. Too little training data could result in low classification accuracy (Manning & Schütze 1999 p. 586).

For IBM Watson NLC, 5-10 data sets (IBM n.d.-b sec. best-practices.html#best-practicesfor-classifiers) are sufficient according to the documentation. A learning curve (Figure 8) has been computed for the IBM Watson Natural Language Classifier (IBM NLC). This learning curve shows that for this particular method, indeed only a very modest number of training set samples per class is needed to reach a useful accuracy: The minimum number of sample texts recommended by the vendor, namely 5, was sufficient to correctly classify over 70% of the texts in the test set.

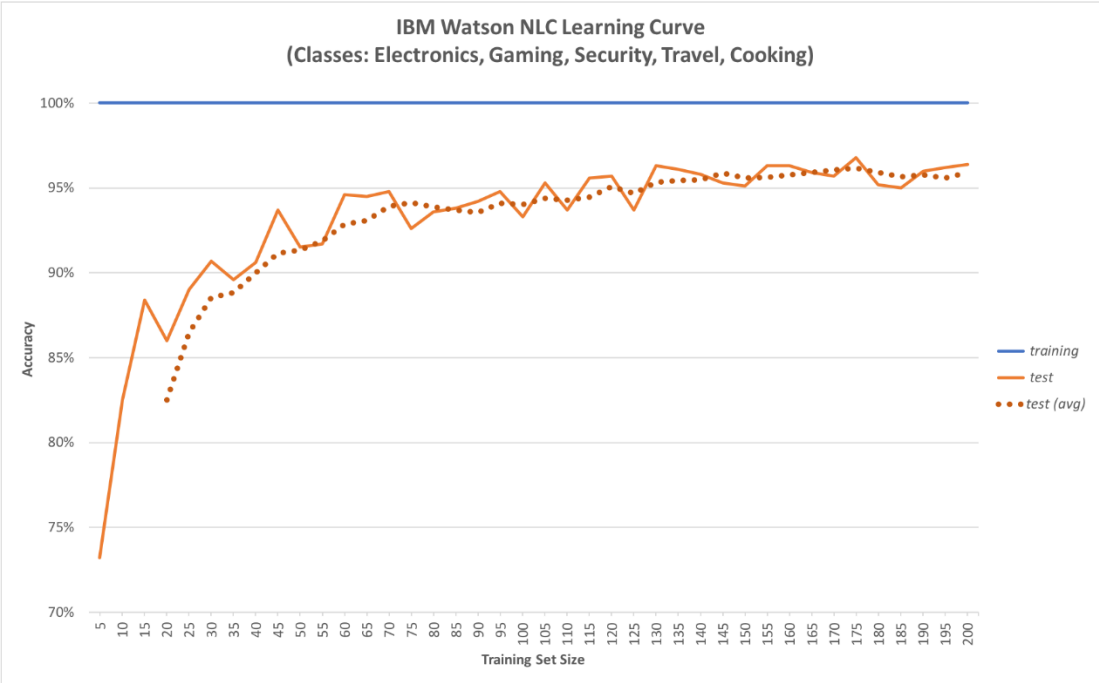

Figure 8: Learning curve for the sample threshold - own computation

How the distribution of classes for the training set will finally look, depends on the occurrence of the classes in the corpus itself. If text labels have been confirmed during the correction phase, or a change to another existing label has been made, the training set should contain these. Through this training set, the new classification model, extended by the new classes detected and confirmed through HITL, can now be created.

Resulting data to be stored / saved:

- Classifier version (ID)

3.1.4 How to build a classifier: an excursion

The classifier is built by learning the properties of the classes from a set of correctly pre-labelled training samples (Feldman & Sanger 2007 p. 70). This is called supervised learning because as the classes for the "members" of this initial sample set are known: "it is as if a "supervisor" has provided these class labels" (Hand 2006 p. 1). A supervised learning algorithm studies the provided learning data set and learns how to classify provided texts into the desired classes (Goodfellow et al. 2016 p. 103) The training set consists of the sample size per class (determined through the learning curve), times the number of classes to be trained.

The training set needs to be prepared in a way that the features required for recognizing the class (Leek 2015 p. 50/51), i.e. the characteristics that make up a class, are included in the entire data set. Furthermore, a test sample set is required to test the classifier for accuracy.

The test set must be structured similarly to the training set regarding the features, but the samples contained should not occur in the training set: A classifier will always perform well on the data it was trained with (Manning & Schütze 1999 p. 577 and Stanton 2013 p. 186). This can be seen in the learning curve - classifying the samples from the training set with the trained classifier results in a 100% accuracy (the blue line on top of the graph). The test set represents the unseen data the classifier will perform on in the future (Manning & Schütze 1999 p. 577). The size of the test data set can be determined individually, but standard set sizes are for example 2/3 training data to 1/3 test data (Stanton 2013 p. 188) or 3/4 to 1/4 (Peng & Matsui 2015 p. 135).

One topic to address here is the problem of how to model the label distribution within the training set. The question is whether to use balanced or imbalanced data in a training set. In an unbalanced distribution, some classes are represented only by a few, and some classes by a lot of samples in the training set (Kao & Poteet 2007 p. 5).

It is difficult to give an informed statement about the label distribution within the training set. This is a topic barely touched in the literature, and mostly discussed in Data Science[6] or Statistics[7] forums.

Some claim that unbalanced training data, modelling the factual distribution of classes in the environment is relevant for the classification accuracy on frequent classes. In Liu et al. (2007 p. 171) on the other hand, it is argued that a classification algorithm can "[...] be over-whelmed by the major categories and ignore the minor ones." An unbalanced label distribution could be presented in a case where infrequent events need to be detected, such as in credit card fraud. Such an event presents an important category but will naturally have only a few samples compared to the available corpus (Liu et al. 2007 p. 171).

According to Forster (2018), training sets should be balanced in order to minimize the risk that a rare case could be ignored. Bakhsh, Mohammad, & Berajawala (2018 p. 17) also mention that class imbalances in the training set could lead to misleading classifier performance metrics for IBM NLC.

The best solution would probably be to build models with both a balanced and an imbalanced data set, if the environment has an underlying skewed label distribution. Both classifier models should then be tested in a test representing the actual distribution. Comparing the classifier performance on both models should give a good foundation for a decision concerning the training set composition.

[6] https://datascience.stackexchange.com/questions/810/should-i-go-for-a-balanced-dataset-or-a-representative-dataset, as an example for a data science forum.

[7] https://stats.stackexchange.com/questions/227088/when-should-i-balance-classes-in-a-training-data-set, as an example for a statistics forum.

Before building the classifier though, it is necessary to determine the corpus from which both the training and test set will be drawn. The labels to be trained must be represented in the corpus. These are usually texts (unstructured data) a from the application respectively the environment (for example, a customer care system or a mail system) to which the classification system is to be applied. This "historical" data, usually already pre-classified by humans or another system, represents the raw data for the classifier to be trained on. As mentioned above, this data must then be cleaned and prepared, depending on the requirements of the method used.

According to Kitchin (2014 p. 105), it's usually either a Naïve Bayes classifier, a decision tree technique, a Neural Network or a support vector machine (SVM) used for classification tasks. The selection very often depends on the use case. In our case, the IBM Watson Natural Language Classifier (IBM NLC) is used, a "ready-made" service that is available as an API in the IBM Cloud. The IBM NLC is currently only able to process texts with less than 1024 characters (IBM n.d.-b sec. using-your-html#datapreparation). With training data appropriately prepared, the classifier (the model) is then trained and if necessary retrained until the desired result is achieved, and the classifier can be used in a "live" application.

In IBM NLC each trained classifier model has its own unique ID, which allows the labelled texts to be assigned to a specific model or version. This provides the system to keep track of which data was classified by which model.

The cleaning and preparing steps are included in the conceptualization as the component "Feature Extraction".

As the process of cleaning and selecting features depends heavily on the technique or algorithm used for classification, we will not discuss the necessary steps here. It is not relevant for this work and would go beyond the scope of the master thesis. Feature selection was touched upon in chapter 2.2.1, page 14.

3.1.5 Step 4: set productive

As soon as the new classifier model has been evaluated, it can then be used productively and replace the old classifier used currently in production for Step 1 (classify). As metrics have to be compared and advantages and drawbacks of the single models have to be balanced against each other, this is probably a manual step where the performance of the model is compared to that of the current model. If the employed algorithm(s) or classification allow so, the old model should be versioned

3.2 Theoretical framework

All relevant hypotheses have now been formulated and based on solid groundwork. The hypotheses framework underlying this master thesis thus can be represented in the following way.

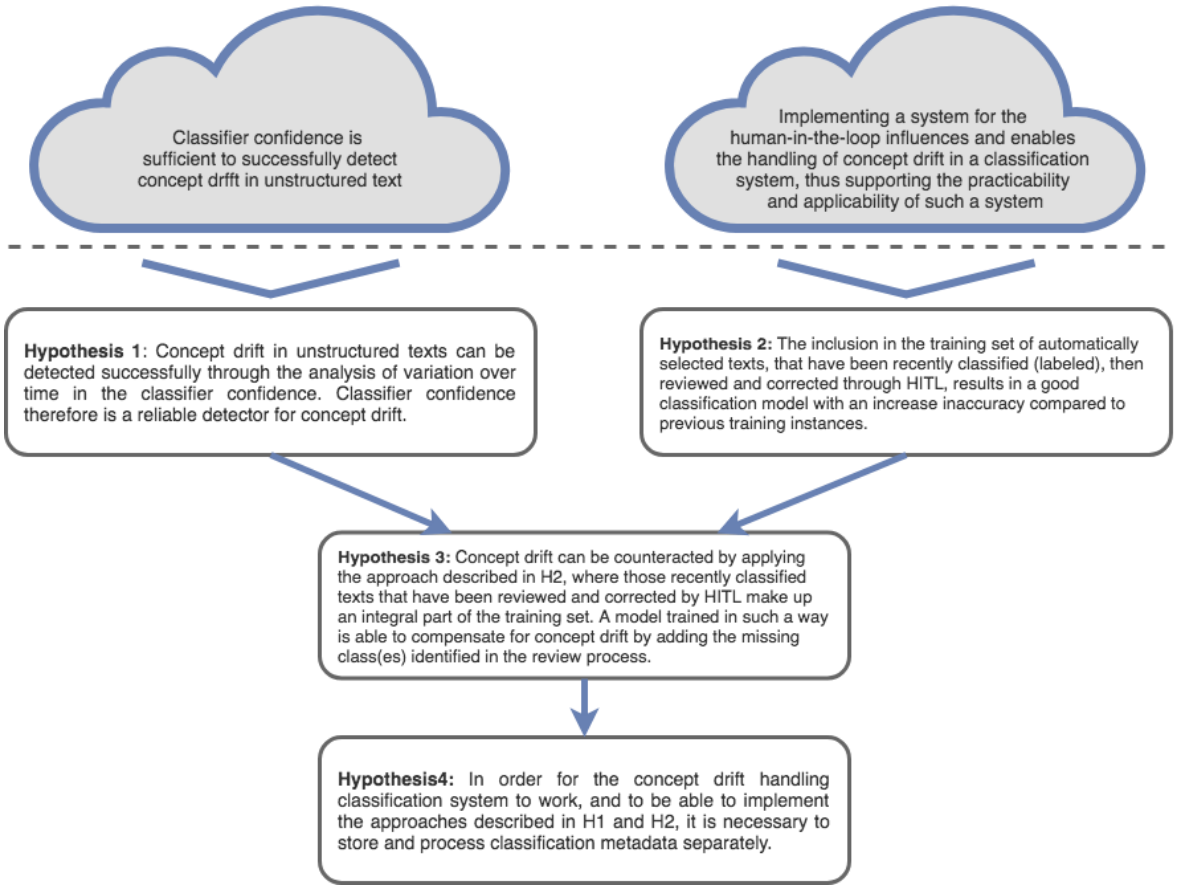

Figure 9: Hypotheses framework with the assumption base

While the clouds on the top represent the assumptions that triggered the development of the system in the first place, the rectangles represent the hypotheses that are the basis for the system to actually work

3.3 Components of the solution

From the problem statement and the resulting solution objectives, the individual steps discussed above, several solution components can be identified. As the problem components in the previous chapter, the solution components should be understood before designing the solution itself. The solution, defined by its objectives, consists of the following components:

A natural language text classification service. Within this component, it is necessary to especially highlight the

- Classifier confidence
- Training requirements (this is already covered in section 3: re-train)
- Applied service: IBM Watson Natural Language Classifier

The Human-in-the-loop

3.3.1 Classifier confidence

As we have seen, a classification model is built by using a training set that correlates the features in a sample to the pre-assigned class label. For a new input object (a text), the trained model then attempts to assign a class based on what the underlying algorithm was able to infer from the data-label correlation evident in the training set. It is a process where the classifier tries then, based on available evidence, to predict which label best represents a given text.

How this assignment is done can be attempted in two ways (Sebastiani 2002 p. 21; Aggarwal & Zhai 2012 p. 164):

- The hard (fully automated) classification assigns a specific label explicitly to a text. For a pair of ⟨dj, ci⟩ the classifier assigns a truth value: if the document d belongs to the class c, then the truth value = TRUE, if not then FALSE. We have seen this on page 11 in the section "A formalized representation".
- In the soft classification, also called a ranking (semi-automated) classification, the classifier returns the so-called Categorization Status Value (CSV), a probability value that reflects / mirrors how confident the classifier is that a text belongs to a specific class.

We will refer to CSV as confidence value or CV for the remainder of the master thesis though, as the abbreviation is most commonly known and used for "comma-separated values", a plain text-type file (this is also the term used in the IBM Watson NLC service documentation, probably to prevent confusion with the input files it requires, which happen to be comma-separated values files).

So, instead of returning a truth value, the classifier produces a confidence value, a number between 0 and 1 (Sebastiani 2002 p. 21) that signals the relevance between a text and a given label. 1 represents the absolute confidence about a positive relationship between a text and a class, while 0 represents absolute confidence that a text does NOT belong to a class. Usually, floating points are returned for each label trained in the model. Labels are then ranked according to their CV value.

In IBM NLC, the classification service API used for this master thesis, the classification output lists class-confidence value pairs in descending order of confidence for each classified text (see Figure 10).

```
{
  "classifier_id": "10D41B-nlc-1",
  "url": "https://gateway.watsonplatform.net/natural-language-classifier/
  "text": "How hot will it be today?",
  "top_class": "temperature",
  "classes": [
    {
      "class_name": "temperature",
      "confidence": 0.9998201258549781
    },
    {
      "class_name": "conditions",
      "confidence": 0.00017987414502176904
    }
  ]
}
```


Figure 10: Classification output of an example IBM NLC classifier.
Source: IBM n.d.-b sec. getting-started.html#natural-language-classifier

The higher the confidence value, the more confident the classifier is that this is the relevant label. All confidence values always sum up to 1 (or 100%). In the example classification output, only two classes were defined, hence only two class - confidence pairs.

As, according to Bakis et al. (2017 p. 2) accuracy is correlated with the classifier confidence, a confidence threshold can be applied in order to determine the correct label. Only that specific label above the threshold represents the correct class and should be assigned to the text. This corresponds to a "binary decision" (Feldman & Sanger 2007 p. 67), with CVs above the threshold representing a TRUE decision, and those below a FALSE decision. Bakis et al. (2017 p. 3) mentions a threshold of about 70% for the confidence to be relevant. As stated before, for this paper a CV below the threshold of 80% is assumed to be too low, requiring a manual intervention.

The CV could also be used as an accuracy measure by computing the confidence average for the assigned class (that class - CV pair with the highest confidence) over the whole test set. The higher that average, the higher the assumed accuracy. An example: if the system returns classifications with confidences around 70-80%, then roughly two-thirds to three-quarters of all texts should have been labelled correctly (Bakis et al. 2017 p. 3).

3.3.2 IBM Watson Natural Language Classifier

For this master thesis, the IBM NLC service is used for the classification task. The decision for this API is mainly based on the fact that the IBM Watson Natural Language Classifier is a

"ready-made" service that is available as an API in the IBM Cloud[8]. It can be used without any background in machine learning or natural language processing. The drawback is that it presents itself as a black box system from the user's perspective and can't be modified.

IBM NLC uses an ensemble of machine learning algorithms (Bakhsh et al. 2018 p. 12) that return a confidence value for all possible classes for a text, thus predicting the most probable label for a given text. The service is pre-trained on Wikipedia with unsupervised training (Bakhsh et al. 2018 p. 11) to provide the word vectors needed for feature selection, as mentioned in chapter 2.2.1. This way, only marginal data preparation for the training set is needed, with no additional feature extraction necessary.

Ensemble means that a combination of ML models is applied to achieve robust results. The idea behind ensemble learning is that, as different models usually have different weaknesses, a combination of multiple models should deliver consistently good results (Aggarwal & Zhai 2012 p. 209).

Information on the exact composition is classified and for IBM internal use only. According to a mail exchange with the product offering manager at IBM (see Appendix 11), underlying parameters and the algorithms used cannot be made public, "exposed". Therefore, no additional information will be provided here.

The service applies deep learning and engages GPUs especially when large training sets are used, usually when a training set size reaches about 15.000 samples (Bakhsh et al. 2018 p. 15). This should enable reasonable training times. Just as an example: A training iteration with 1585 samples for 5 classes took 57 minutes[9].

Unfortunately, as mentioned above, no information is available as to how the confidence value is computed and assigned.

3.3.3 The human-in-the-loop

The Human-in-the-loop (HITL), can be defined, simply put, as a model of something that requires human intervention or interaction.

Recent literature mainly associates the HITL model with two subfields of artificial intelligence (AI) though: robotics (e.g. Nunes, Zhang, & Silva 2015; Schirner, Erdogmus, Chowdhury, & Padir 2013) and machine learning (e.g. (Amershi, Cakmak, Knox, & Kulesza 2014; Holzinger 2016; Manning et al. 2009)). So, while many AI endeavors aim to remove the human out of the loop, such as in the field of autonomous driving (Holzinger 2016 p. 120), the HITL researchers attempt to add the human back into the loop. For example, in order to reduce

[8] As the author worked for IBM at the time of writing the master thesis, it was also the most convenient choice of classification service.

[9] Training on 24/07/2018

complexity through human input, such as knowledge of a domain expert into a system with high uncertainty or complex patterns (Holzinger 2016 p. 119 and 120). One such domain could be diagnostic radiologic imaging (Holzinger 2016 p. 122).

Especially in the field of ML, this is not necessarily called a HITL-approach, but is known by other terms respectively operating principles, specifically interactive, active and coactive learning (Holzinger 2016 p. 124, Gurevych et al. 2017 p. 3) .

In robotics, it can be referred to as HiTLCPS (Human-in-the-Loop cyber-physical system) (Nunes et al. 2015 p. 946) or HiLCPS (Schirner et al. 2013 p. 6). But regardless of what it is called, the human component is a central aspect.

From this perspective, we could define HITL as an interaction between machines or algorithms and human agents.

Holzinger (2016 p. 120) defines the interactive ML approach as "algorithms than can interact with both computational agents and human agents and can optimize their learning behavior through these interactions." An algorithm (a learner) can incrementally update the model based on the feedback and the actions of the agent (both human and machine) (Gurevych et al. 2017 p. 3). One could say that the system stays under construction while it is used, getting better with time and utilization.

Active learning is similar, but with a strong focus on the labelling aspect (Gurevych et al. 2017 p. 3), where the system asks the agent (the human) for selected labels, usually those with the lowest confidence (Žliobaitė et al. 2014 p. 27), in order to increase accuracy (Žliobaitė 2010 p. 16). But the agent has no influence on the model learned from the manually labelled data, other than the labels provided (Gurevych et al. 2017 p. 3). Coactive learning on the other hand gives the agent, the user, more influence and can be viewed according to Shivaswamy & Joachims (2015 p. 2) as "[...] a cooperative learning process between system and user, where both parties aim to optimize utility but lack the means to achieve this goal on their own."

From this we can deduct that a HITL approach is usually chosen when a purely automatic approach (in terms of machine learning) would be insufficient, just not feasible or if the input of an external agent could improve the outcome. This could be the case e.g. in the health domain, where data sets are sometimes small or rare events represent important labels, which would lead to insufficient training samples (Holzinger 2016 p. 119).

In the field of robotics, the human-in-the-loop represents a significant part of a system, where the complete system augments and improves the interaction of the human with the world around him (Schirner et al. 2013 p. 36). An example of such an HITL in a cyber-physical system would be an intelligent prosthesis (ibid.).

According to Munir, Stankovic, Liang, & Lin (2013 p. 1), HITL applications can be organized into three general categories:

1. applications where humans directly control the system (human control)
2. applications where the system passively monitors humans and takes appropriate actions (human monitoring) and
3. a hybrid of 1 and 2 (hybrid systems)

Applications where a human directly controls a cyber-physical system are categorized under human control. Here the system executes the orders given by the human. An example would be a robotic arm. Human monitoring comprises applications that passively monitor humans, such as activity trackers do. Hybrid systems take the sensor input from the monitoring aspect as feedback for a controlled system, replacing the commands given by the user with the monitor feedback. This could be represented by an activity tracker with a heart rate monitor, that sends a warning signal if the heart rate is too high (description according to Nunes et al. 2015 p. 954).

The HITL approach used in this paper roughly falls under active learning, as the human agent is called in cases with high uncertainty, i.e. low confidence labels. It is not the ML algorithm itself that requests feedback from the user, but the system encompassing the algorithm, which is why it is ultimately not an active learning approach. The system acting as middleman between classification service and the HITL though provides both the application and the user with more freedom as to

a) Decide which labels will be corrected, because the feedback is not needed for the classification and
b) Adjust the confidence threshold as needed, without having to interact with the algorithm itself

4 Designing the solution: a high-level architecture

The goal of this chapter is to provide a high-level architecture for the system conceptualization (Figure 7), that would enable and facilitate the implementation of such a classification system. An architecture is, according to the Oxford Dictionaries, generally speaking "the complex or carefully designed structure of something."[10] In order to get from the objectives described in the previous chapter to a structured approach on how to implement the proposed classification system, it is necessary to describe the system in an organized way following architectural conventions.

According to Cook, Cripps, & Spaas (2008 p. 5), it is impossible to create a comprehensive model of a system that captures all functionality and quality properties in one single representation, especially if it needs to be understood by all stakeholders. It is therefore necessary to break the system model down into separate but related views that "[...] collectively, describe the whole system."(ibid.)

A view would be a representation of a part of the architecture that relates to a specific viewpoint on the system. A viewpoint is "[a] specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis." (Cook et al. 2008 p. 3).

A system can be looked at from three different viewpoints, thus focusing on fundamental aspects (Cook et al. 2008 p. 6 and 7):

- **Requirements**: Representations that capture all requirements placed on a system such as business, technical, functional and non-functional requirements and constraints.
- **Solution**: Representations modelling the solution that meets the requirements. Solution models and patterns can be either functional or operational. Functional models are focusing on modelling the system components as well as their relationships. Operational models are representing how the system is built from the description in the functional view.
- **Validation**: Models that are assisting in assessing if a system will deliver the functionalities described in the solution view, with the expected quality described in the requirements view.

This chapter focuses on the requirements and the functional solution viewpoint.

[10] Definition No. 2 of "architecture". https://en.oxforddictionaries.com/definition/architecture [28/07/2018]

Validation models are strongly related to the actual environment and its requirements in which the application is to be implemented. Assessing whether the application will deliver with a specific quality therefore would be a task for the architect actually implementing the system. This is in part due to the fact that especially the non-functional requirements (NFR) and the constraints depend strongly on the given operational environment, which of course cannot be taken into consideration in this master thesis.

In the remainder of this chapter we will therefore discuss some aspects of the requirements viewpoint by:

- Illustrating the environment, in which the application has to operate through the system context diagram. A system context diagram identifies external interfaces to both human users and other IT systems (IBM 2018b p. 24).
- Listing the functional (and, where possible non-functional) requirements, as well as constraints.
  - Functional requirements (FR) specify what the system should actually do, for application users to be able to fulfil their job, for example "store data".
  - Non-functional requirements (NFR) define the characteristics and expectations placed on the system. They describe how a system works, e.g. by defining an expected response time.
  - Constraints describe given facts that cannot be changed for the project, e.g. the existing infrastructure making up the environment of the system. Other constraints could be budget or skills available for implementation.
- Creation of the UML use case model for the proposed classification system. Use cases define particular actions or goals a user wants to achieve by using the system. The use case model would then be the summary of all use cases within the scope of the system (IBM 2018b p. 30).

The functional solution viewpoint is represented by

- The flow model which depicts, according to (Hartson & Pyla 2012 p. 133) the overview of work roles and system components involved, as well as the communication and information flows between them.
- The component model, which represents the internal structure of the system (IBM 2018a p. 5). In this case, it also functions as the architecture overview to provide a high-level outlook on the proposed system architecture. It is supposed to communicate a conceptual understanding to the stakeholders.
- as well as four UML sequence diagrams, that show how the components interact in order to satisfy the use cases.

4.1 The system context

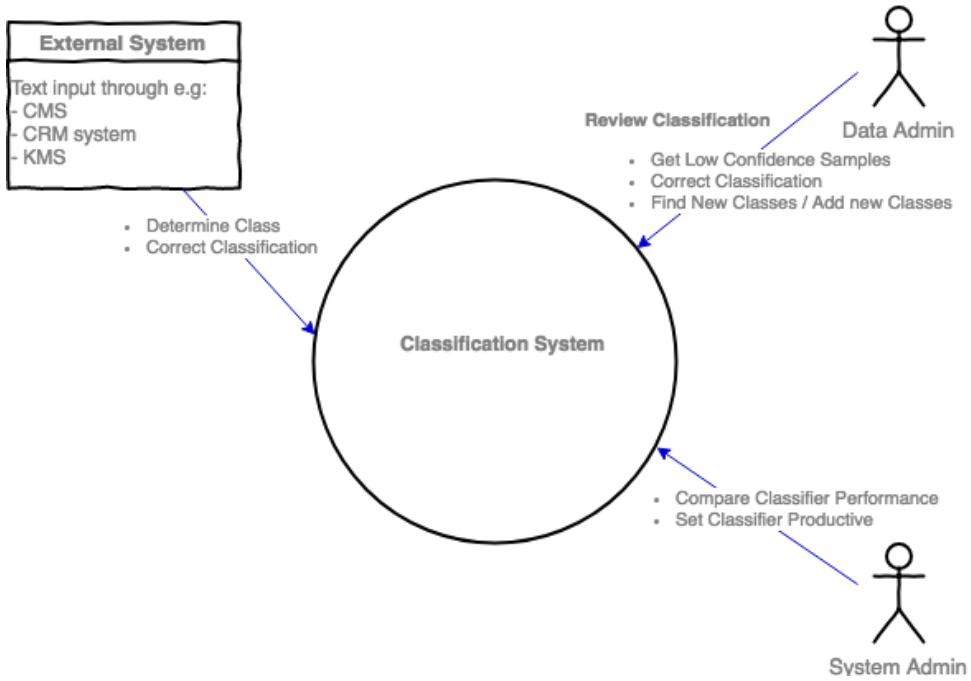

Figure 11: System context diagram (own depiction)

The system context diagram illustrates the interfaces to external systems, human or IT, while depicting the system itself as a black box. It is used to identify all actors interacting with the system. For the text classification system, three actors were identified:

- The external system that provides the texts to be classified, and which expects to retrieve a classification label. In the case of a revised classification by the HITL, the corrected label needs to be played back into the external system. Such a system could be e.g. a knowledge management system (KMS) or a customer relationship management (CRM) system.
- The data admin is the central human-in-the-loop. The HITL receives the low confidence samples and plays the revised classification back into the system. New classes are discovered here. The data admin most probably is some kind of domain expert or advisor for that specific field.
- The system admin is responsible for evaluating the newly trained classifier models, and to set them productive. The system admin needs to have some expert knowledge in the field of classifier evaluation, most probably someone with a background in statistics or data science.

The context diagram defines the boundaries of the system and how it needs to interact with external actors. It acts as a base for the definition of the functional requirements.

4.2 Functional requirements

The system context diagram identifies high-level requirements placed on the system from outside. These can be considered the "nouns and verbs" that describe what the system does (IBM 2018b p. 35). Since Functional Requirements specify what the system should do, the question needs to be asked what the system actually will be doing, considering the external requirements. The high-level requirements from the system context as well as the objectives from the system conceptualization therefore need to be itemized, and are listed below according to the steps from chapter 3.1, "System conceptualization".

Step 1: classify

- The system must accept unstructured data in text form
- Labelled texts must be sent back to the external system
- The system must prepare the data according to the classifier needs (e.g. strip some punctuation and special characters)
- The system must classify said unstructured data into given classes, using a pre-trained classifier
- The system must store all classification meta data

Step 2: correct

- The system must send to HITL
  - Texts with a classification confidence lower than the defined threshold
  - Plus labels and their confidence
  - Plus already existing, but not yet trained new classes
- The system must accept corrected and confirmed classes (existing & trained / existing & un-trained / new)
- The system must keep track of those texts that were reviewed (texts with low confidence, where the class was corrected or confirmed by HITL) to feed them into the next re-training step
- The system must keep track of classes
- The system must keep track of number of texts per class

Step 3: re-train

- The system must prepare training sets for new classifier training automatically when:

  The sample threshold for new classes is reached. A training set is only prepared, if:

  1. A sufficient number of samples for training are available (this is a pre-defined threshold)
  2. Additionally, enough samples to create the test set are available (this is the pre-defined test set size, identical for all classes)

  The training set must contain:

  1. Enough samples from all classes (the sample size depends on the pre-defined sample threshold)
  2. The last reviewed texts, i.e. texts for which the class has been corrected or confirmed by HITL for both existing and newly identified classes

- The system must prepare a test set for new classifier training from random texts from all classes previously not included in the training set
- The system must automatically train a new classifier
- The system must test new classifier with said test set

Step 4: set productive

- The system must make evaluation results available for HITL
- If evaluation results are satisfactory: The system must accept HITL's "set-productive" decision, and switch the model in production (i.e. make them available for the classify step)

4.3 Non-functional requirements

NFR can be seen as a kind of "adjectives" that further specify the nouns and verbs representing the functional requirements (IBM 2018b p. 35). As they depend strongly on the given environment, only a few assumptions can be made concerning the NFR. In a real-world scenario, they are dependent on the implementation scenario. Here are exemplary NFRs that could be relevant for an implementation:

- Latency: The response time of the classifier must support the type of application that relies on it, e.g. for use in an internet forum, the classification must be finished by the time the user is finished writing his post (within seconds). Other applications have more relaxed requirements, e.g. adding a document to a document management

system is an asynchronous operation, meaning the classifiers' response time doesn't have to be low enough to support real-time operations.

- Number of classes: The classifier needs to be able to support the existing number of categories present in the application. If there are more classes existing than are supported by the classifier, a classifier chain might be necessary to be built.
- Availability and Time to Recovery: If the classifier becomes a necessary component of the external system, its availability characteristics will impact that system's availability. It must therefore be designed in a way to provide sufficient availability (redundant design where needed, or a recovery scenario).
- Capacity / Scalability: The system must be able to support the external system's expected peak load. If the load fluctuates in time (e.g. in vs. outside of office hours), a serverless or on-demand solution may limit costs significantly.

4.4 Constraints

Like NFR, constraints depend heavily on the given context and environment in a real-world scenario. They represent limitations put on a solution from either the business side or the IT department. They describe facts that can't be changed for the project. For the classification system the following could be possible constraints:

- Classifier constraints such as max characters, number of texts per batch, number of classes, depending on the classifier engine or algorithm employed
- Constraints depending on specific use case scenario such as code language, infra-structure such as the database type, or legislative restrictions such as GDPR[11].
- Security constraints: Depending on the data being classified, security standards or regulatory requirements may pose specific demands on the system. This could be e.g. the requirement of encrypting data in transit or at rest if it is stored by the system.
- IT constraints such as the existing infrastructure, respectively the environment in which the application is supposed to run in (cloud, container-based, etc), or which type of API is supported by the infrastructure (REST or SOAP[12])

[11] General Data Protection Regulation: a legal framework that sets guidelines for the collection and processing of personal information of individuals within the European Union (EU). Definition from https://www.investopedia.com/terms/g/general-data-protection-regulation-gdpr.asp [22/07/2018]

[12] Both are web service communication protocols. Definition from https://stackify.com/soap-vs-rest/ [02/08/2008]

4.5 Use case model

Use cases (UC) model how users, also called actors, use a system to achieve specific goals. Based on the context model and the system objectives, seven use cases and three actors could be identified for the scope of the system. The use cases "review classification" and "compare classifier performance" represent super-classes.

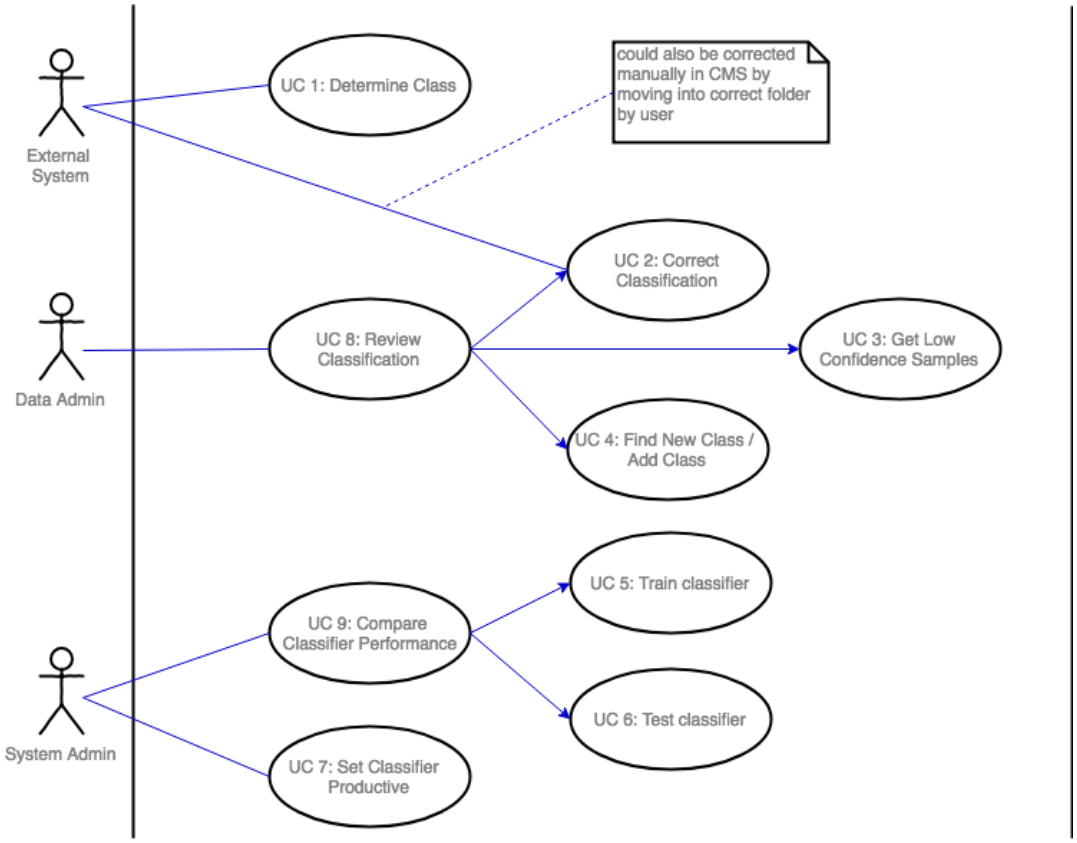

Figure 12: Use case model (own depiction)

UC 1: Determine class: The external system needs the classification system to determine the class of a new text

UC 2: Correct classification: If the initial classification is erroneous, the classification needs to be corrected and played back into the external system. The data admin corrects falsely classified texts and plays them back into the system. This can be done by HITL when reviewing classifications, or by the users of the External System (e.g. by changing the classification in that system).

UC 3: Get low confidence samples: In order to be able to assist the system with improving the accuracy and detecting concept drift, HITL needs low confidence samples to be accessible: A list of all low-confidence samples is generated by the System.

UC 4: Find new class / add class: HITL detects new labels in the texts presented to him, and plays them back into the system

UC 5: Train classifier: When necessary, the system admin requests the system to train a new classifier model to later replace the old, outdated classifier model

UC 6: Test classifier: The newly trained classifier model needs to be tested against a test set, and classification results made available for the system admin.

UC 7: Set classifier productive: The system admin evaluates the new classifier model against the current productive one, and sets the new model productive, in case of a positive review.

UC 8: Review Classifications: Periodically, HITL reviews and corrects classifications by first displaying all samples that have a low confidence rating (UC 3), then correcting or confirming the classification on these samples (UC 2). During this review, HITL may identify a new class in these low-confidence samples, which is then added to the list of classes (UC 4).

UC 9: Compare Classifier Performance: After a review has been executed, HITL may request a new Classifier to be trained (UC 5). This is in particular useful if the number of samples for a newly identified class has increased because of the review and is now over the threshold needed to train a new class. Once this classifier is trained, it is run against a test set (UC 6), and its quality examined by HITL.

4.6 Flow model

The flow model combines the system context view with the use cases, presenting an overview of how the identified actors and possible system components interact with each other. Communication and information flows, as designed for the target state, are also depicted.

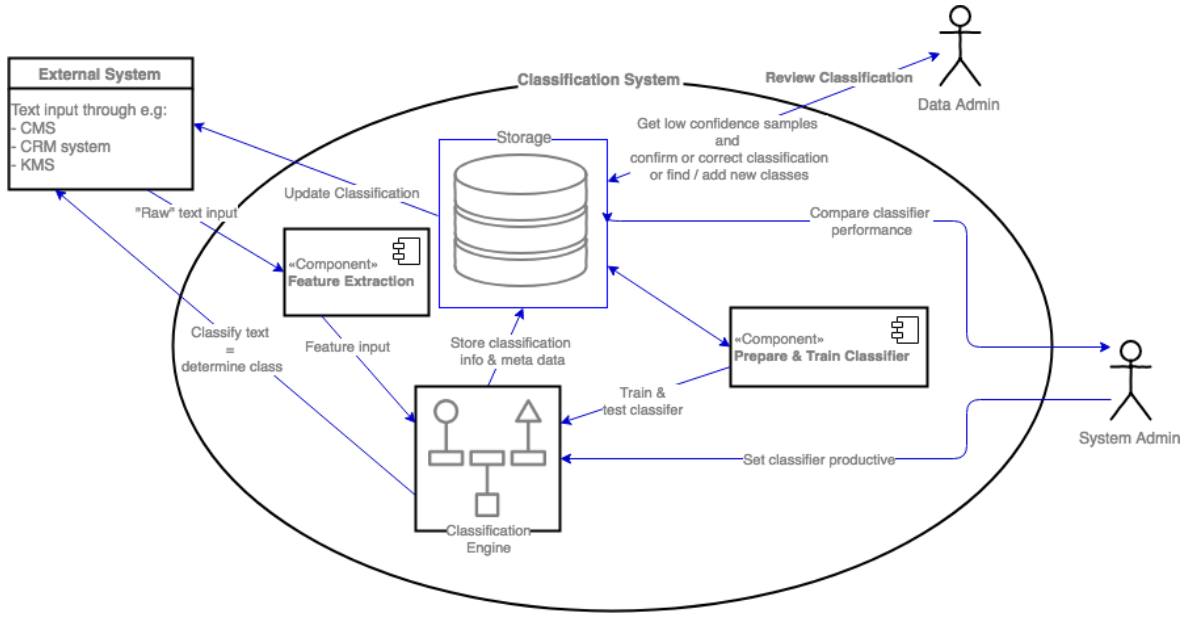

Figure 13: Flow model (own depiction)

While the importance of the storage aspect was already presumed in the previous chapter, it is now possible to see the integral part it plays for the system. All components and actors need some kind of interaction with the storage component.

4.7 Component model

The component model is a formal representation of the relationships between the components that make up the solution (IBM 2018a p. 5). It provides necessary input for the development of more detailed models, as various aspects about the components are now properly defined (IBM 2018a p. 5), such as the number of tiers the system has, or where the components are supposed to run, i.e. inhouse or in the cloud.

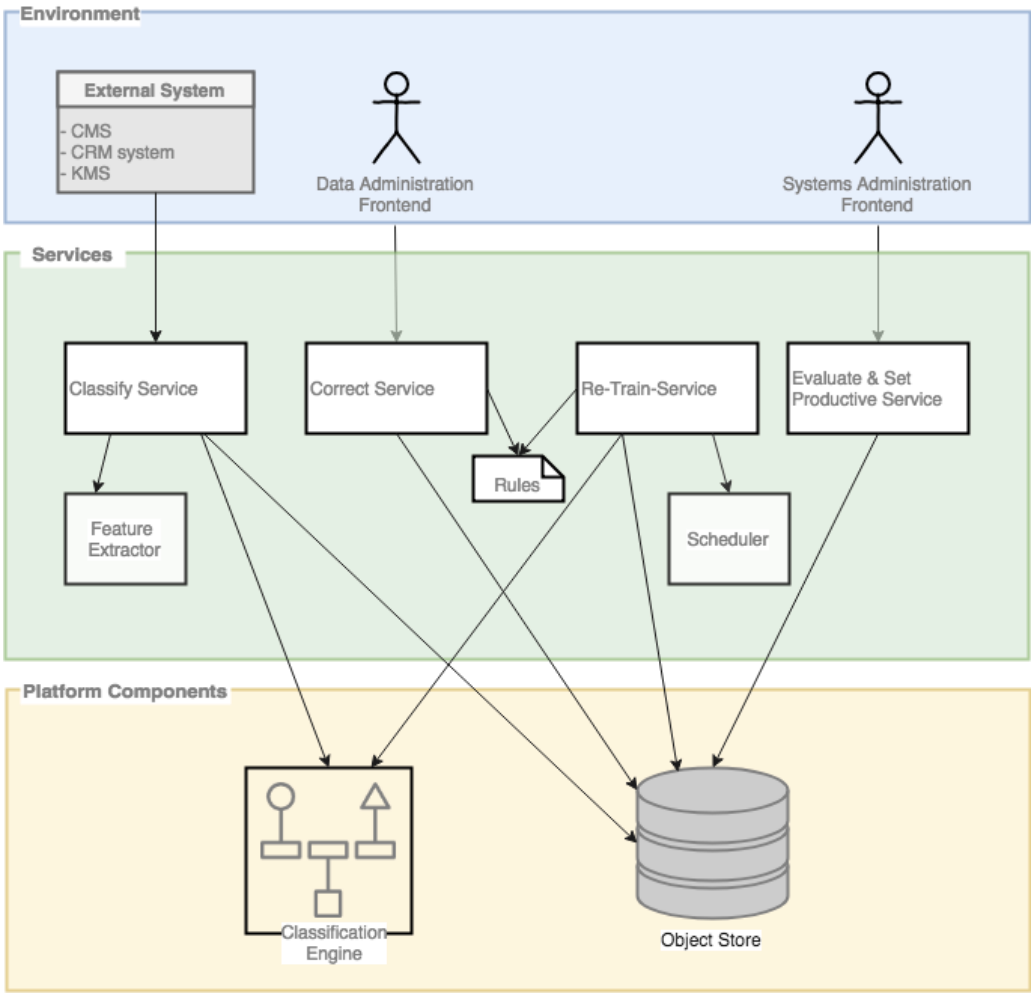

Figure 14: Component model / architecture overview (own depiction)

In addition to the three actors interacting with the system, six components and one rule-set have now been properly defined. The *Feature Extractor* is a sub-service for the *Classify Service* and has no other interaction than with the *Classify Service*. The *Scheduler* is a sub-service for the *Re-Train Service* but could be used for other services if needed. Both the *Correct* and *Re-train Service* access the rule-set, most probably a simple text or .csv file, which contains the confidence and the sample thresholds as described in chapter 2 in the sections "Step 2: correct" and "Step 3: re-train", as well as the sample set size. The decision to treat the ruleset as a separate component instead of integrating it into the corresponding services was made in order to make it more maintainable. By separating the ruleset and not hardcoding the thresholds into the services, an adaption to the thresholds is very easy, and no program code needs to be changed. In addition to that, it allows for other services to also access the thresholds without the need for duplicating it across components.

4.8 Sequence diagrams

Four sequence diagrams that show the interactions between the single components were developed in UML notation. They represent the Use Cases 1, 2, 3 and 5. UC 4 has no component interaction that differs from UC 2, as the decision for new labels solely happens within the Data Admin actor. The "Test classifier" is similar to the "Train classifier" use case concerning the component interactions, and UC 7 (“Set classifier productive”) is out of scope for this paper, as defined in chapter 3.1 on page 27.

4.8.1 UC 1: determine class

The external system calls the *Classify Service* to assign a class label to a text. The *Classify Service* calls the *Feature Extractor* and uses the features in the call to the *Classifier Engine*, which returns a number of labels, all with confidence scores. The *Classify Service* then calls the object storage in order to store the relevant data for later access.

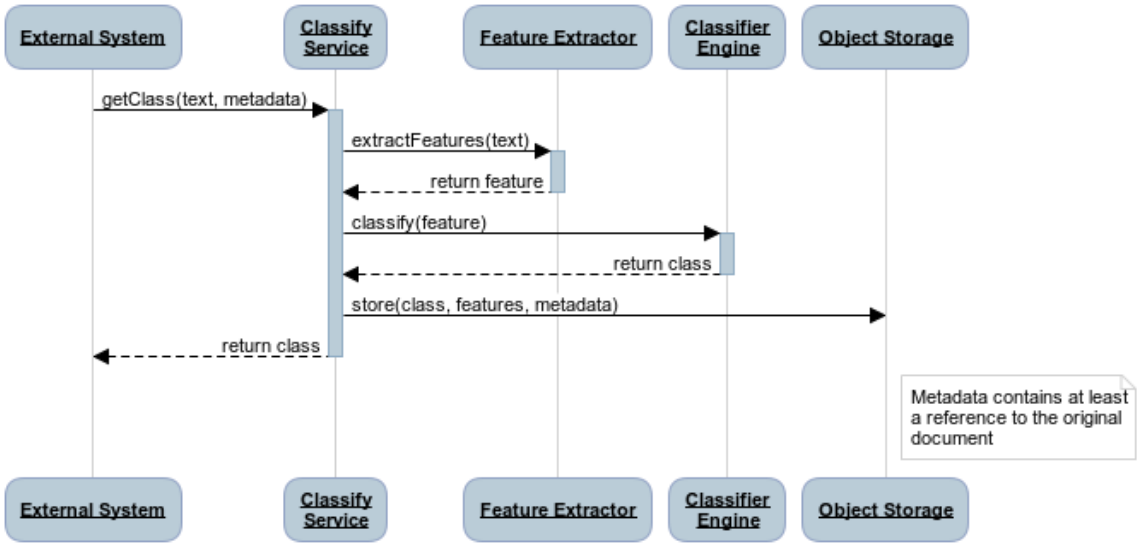

Figure 15: Determine class sequence diagram (own depiction)

4.8.2 UC 2: correct classification

The data admin reviews uncertain labels in the data admin frontend and submits the reviewed samples to the *Correct Service*. The *Correct Service* loops through each item in the submitted set and looks for class updates. If the newly assigned class exists, the update is stored in the object storage, if not, added. If the new class is missing in the object storage, the new class can optionally be created in the external system by e.g. creating a new folder or topic in the external system, and the respective text then filed in the new folder. This could be necessary because the new class might have to be mapped in the external system as a new forum or a SharePoint page. A new class may have a significant impact on its environment.

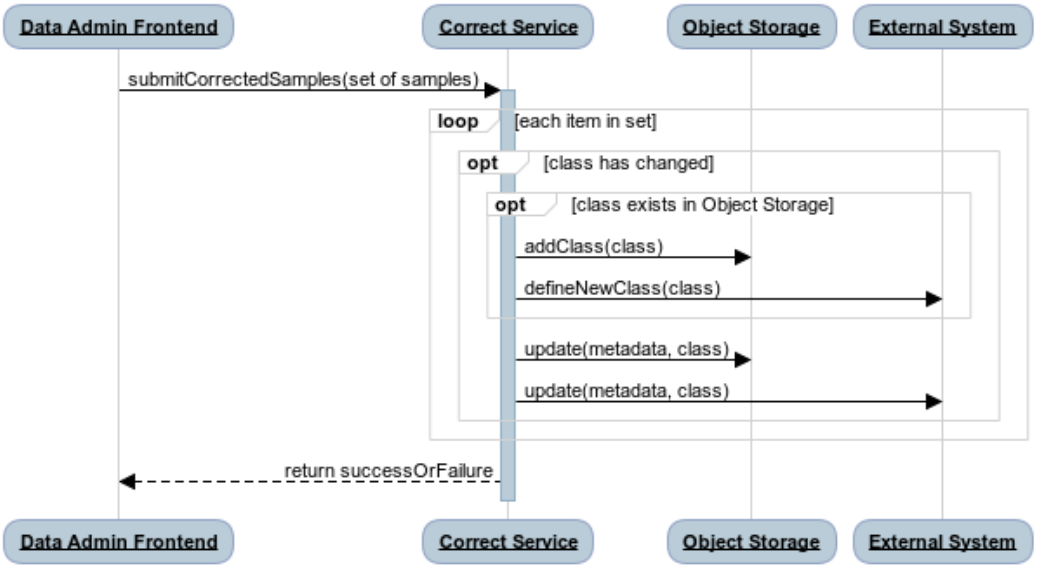

Figure 16: Correct classification sequence diagram (own depiction)

4.8.3 UC 3: get low confidence classes

The data admin frontend requests a list of low-confidence samples from the *Correct Service*, which are typically then reviewed by HITL. The *Correct Service* retrieves the current confidence threshold and checks that against the highest confidence scores for newly classified texts in the object storage. Samples with a confidence below that threshold are then returned to the data admin frontend, which then displays this set of samples to HITL.

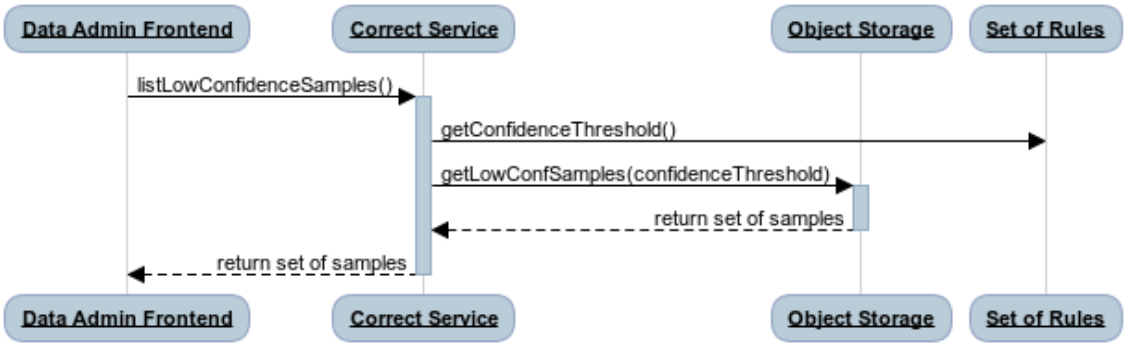

Figure 17: Get low confidence classes sequence diagram (own depiction)

4.8.4 UC 5: re-train classifier

A scheduler, which could be either a human or a machine agent, calls the *Re-train Service*, which retrieves the current sample threshold from the *Set of Rules* and retrieves an according number of samples. If the number of classes to be trained has changed (i.e. a new class not previously trained is existing), the number of existing samples with the new labels is checked against the sample threshold. If enough samples for a training are available, the new class is included in the label set, and the *Re-train Service* retrieves the according

number of feature sets per class from the object storage. The training set is built according to the classifier requirements, and then sent to the *Classifier Engine* for training. After successfully completing the training, the resulting classifier metadata are stored in the object storage for further reference.

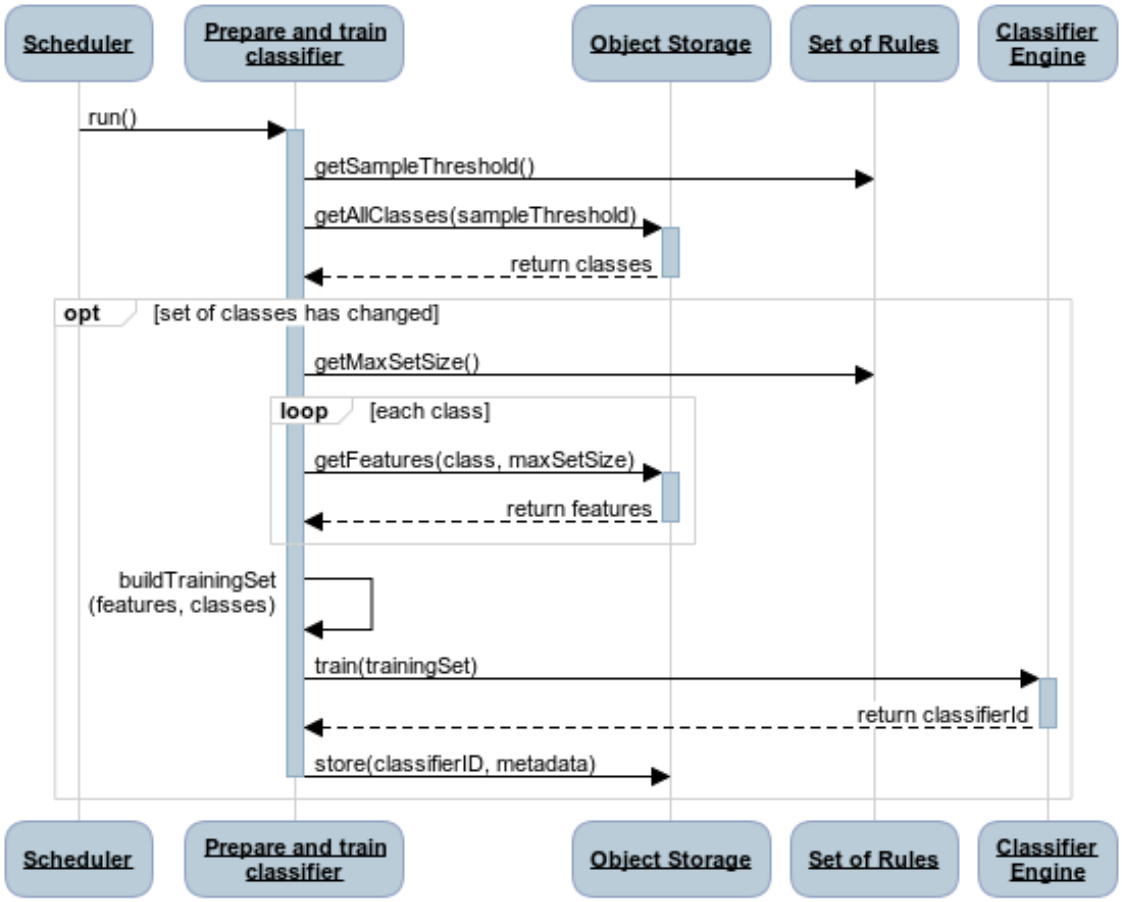

Figure 18: Re-train classifier sequence diagram (own depiction)

In this chapter, the architecture for the proposed system was developed, discussed and presented. It should provide a solid starting point for a practitioner wanting to implement the system. The high-level view was kept in order to as to not impose any unnecessary restrictions.

5 Demonstrating the usefulness of the solution

The goal of this chapter is to demonstrate that the system described in detail in the chapters 3.1 and 4 is able to solve the underlying problem of concept drift through a proof-of-concept (PoC). According to Offermann, Levina, Schönherr, & Bub (2009 p. 5), the evaluation of the soundness of the approach can be achieved by "[...] laboratory experiments or simulations".

Due to the scale and scope of the master thesis, the decision was made to try to verify the hypotheses by conducting two experiments and refraining from building an actual system to mirror the present architecture. The time and manpower needed for this would have greatly exceeded the time allocated to the Master thesis.

A PoC can be viewed as a type of milestone in a project demonstrating the principal feasibility of the project (Rat für Forschung und Technologieentwicklung 2013 p. 2), and to verify that concepts or theories from within the project can actually be achieved. A PoC can be defined as some type of prototype that "[...] is designed to determine feasibility, but does not represent deliverables."[13] It is sometimes also referred to as "Proof-of-Principle" (Rat für Forschung und Technologieentwicklung 2013 p. 2).

According to IBM (2016), a PoC is usually developed early in a project. It can take many forms, such as e.g.

- a list of known technologies such as frameworks or patterns that could be appropriate for the solution or
- a simulation of the solution

Proof-of-Concepts may follow different kinds of goals and can be characterized according to these goals. A User Experience (UX) PoC, for example could aim to gather feedback from a representative set of users, and as such implements a significant part of the User Interface (UI) but may be missing some or all of the backend implementation, as would be a T-Prototype. Other kinds of PoCs could validate other aspects of the solution, such as the interaction with external systems, or how fit the implemented process is for the intended purpose (IBM 2016). Specific parts of the system will thus be implemented accordingly, while others will be left out.

This PoC seeks to validate a method, namely detecting concept drift using confidence metrics, and reinforcing class labels by using corrected low-confidence samples for re-training. In order to achieve this, key aspects of the system have been implemented:

[13] Definition of PoC. https://www.techopedia.com/definition/4066/proof-of-concept-poc [27/07/2018]

- Training the classifier using a commercially available natural language classifier service (IBM NLC)
- Identifying the low-confidence samples
- Reviewing and correcting those samples (automatically instead of manually through HITL)
- Iteratively re-training the classifier using these reviewed and samples

Other aspects have been left out, notably:

- The entire User Interface used by HITL (displaying the low-confidence samples, allowing entry of corrected classes, requesting re-training)
- The integration to an external system (Extracting text from it, sending identified classes to it, updating class assignments through refiling)
- The management of classifiers: Which one is productive at any given point in time, or how newly trained classifiers are set productive.

This PoC consists of two parts:

- A conceptual architecture framework for practitioners, available to be used as a starting point for an implementation in a business environment. The solution architecture has been presented in chapter 4.
- Two experiments to validate the underlying hypothesis, and thus the method. These will be discussed on the following pages.

5.1 The experiments

The raw data used in the experiments described below come from the Stack Exchange[14] forums. Stack Exchange, Inc. regularly makes all contributions of users of the Stack Exchange Network anonymously available as data dump. Each forum is provided as a separate archive. These archives can be accessed free of charge, they are provided under a Creative Commons license.

A large part of the code for the experiments was written by F. Auberson (2018) together with the author, using the authors instructions and guidelines, thus translating the authors specifications into code. The complete code to both experiments can be found in Github under https://github.com/KSAub/MT2018_PoC_ConceptDrift.

[14] Stack Exchange Data Dump dated 13.03.2018 - https://archive.org/details/stackexchange, [18.03.2018]

5.2 Experiment 1

The first experiment is designed to simulate the human-in-the-loop approach for the classification reviews without emerging new classes. The goal was to test *Hypotheses 2,* the assumption that he automatic selection of samples from recently classified (labelled) texts which were reviewed by HITL results in a good classification model, with an increase in accuracy compared to previous training instances.

During the experiment, a classifier model is first trained on a training set composed of 150 random samples of text from five different Stack Exchange forums (Electronics, Gaming, Security, Travel and Cooking). They were selected based on the number of entries in them (more than 10.000 in each of them), which provided enough samples for many experiment iterations. After the training, a set of 600 random samples per forum, which represents the class label, was then classified through the IBM NLC.

From the classification results, those samples with a confidence value below the threshold were selected, automatically compared to their original label and thus corrected. This models the HITL behavior - the HITL selects all samples with a low confidence (confidence threshold 80%). An ideal result of his work is assumed: in the review, the task of reviewing the low confidence samples and assigning the correct classes, the HITL always assigns the correct class.

Those reviewed samples were then selected for the next training iteration and used for the training set. This resembles a sliding window approach, were a classifier is trained on the most recent supervised samples. In cases were the reviewed samples weren't enough to reach the sample threshold, random samples were again drawn from the respective forums. The experiment was iterated 10 times. The expectation is to see an increasing accuracy over the 10 iterations. A graphical representation of experiment 1 can be found in the Appendix, Figure 24.

5.3 Experiment 2

The second experiment is designed to simulate emerging concept drift by adding samples from previously unknown classes and test how the system modeled in experiment 2 would adapt to it. The goal was to test *Hypothesis 1* and *3*, the assumptions that

a) concept drift could be detected by analysis of variation in the classifier confidence and

b) that by applying HITL to detect emerging new classes, concept drift could be countered.

Experiment 2 also starts with the same initial training set of 150 random samples from the classes mentioned in Experiment 1. After the training though, in addition to the 600 samples per class, another 600 random samples from a new forum (here: the Pets forum) were classified, thus introducing concept drift, specifically the "novel class appearance" (see chapter 2.2.2,"Types of concept drift").

The remainder of this experiment is similar to Experiment 1, with the exception of the identification of the new class through HITL. The expectation for this experiment is to see a recognizable dip in accuracy and average CV after the first classifying (testing) iteration. Once the concept drift was detected though, accuracy should soon reach a level compared to that in Experiment 1. The graphical representation of experiment 2 can also be found in the Appendix, Figure 25.

A possible variation of this experiment could be to introduce an error rate in the review, since the HITL will occasionally be wrong about the classification, as manual classification usually does not have an accuracy of 100% (Sebastiani 2002 p. 47).

Another variant of the concept drift experiment would be to define lower and upper bounds of the threshold, in order to enable the system to learn the new class at a much earlier stage. The accuracy would not be too good in the beginning, depending on the accuracy measure at the lower bound, which could be gleaned from the learning curve. The system would be able to classify the new label much earlier, which is a tradeoff to be carefully weighed.

6 Evaluation

In order to be able to test the hypotheses, and thus validate the underlying assumptions of the presented solution, the conducted experiments need to be evaluated. Both experiments are centered around classifier performance, as described in the previous chapter.

Evaluation, within the scope of this paper, can therefore only be done through typical evaluation methods for the kind of algorithms applied in the system and the experiments. The experiments are therefore evaluated using the usual metrics for classification tasks.

The effectiveness or performance of a classifier is usually measured using Precision, Recall, F-measure and Accuracy (Sebastiani 2002 p. 37; POWERS 2011 p. 1; Carrillo, Brodersen, & Castellanos 2014 p. 2; Manning et al. 2009 p. 280).

Precision shows the percentage of how many of the predicted positive samples are in fact actual positives. *Recall* on the other hand shows the percentage of the actual positive cases that were predicted as positive. The *F-measure*, also known as F1 score or F-score, "can be interpreted as a weighted average of the Precision and Recall, where an F1 score reaches its best value at 1 and worst score at 0."[15]

Accuracy is an overall measure on how often the classifier predicted the correct value, and it returns the fraction of samples that are correctly classified. In contrast to the *precision*, *recall* and *F-measure*, which are best suited for binary classification (Hossin & Sulaiman 2015 p. 3), *accuracy* is also suitable for evaluating multi-class classifications (Carrillo et al. 2014 p. 4). For a multi-class classifier, *accuracy* is computed by summing all correct predictions across all classes divided by the number of all samples (ibid.)

A graphical method to "[...] quantify classifier performance is the confusion matrix, which captures counts of how many questions (which belong to class i) have been answered correctly, and how many have been assigned incorrectly to each class j." (Bakis et al. 2017 p. 7) The confusion matrix is also referenced as the "contingency table" (Manning et al. 2009; POWERS 2011; Sebastiani 2002). According to Bakis et al. (2017 p. 7), an accurate classifier should lead to a mostly diagonal confusion matrix, showing that the majority of the samples have been labeled correctly. In addition to that, it allows to identify those pairs of classes that are notably overlapping and are often confused with each other, resulting in larger numbers in the "off-diagonal entr[ies]" (ibid.).

Sebastiani (2002 p. 37) notes though, that class-related (corresponding to TRUE / FALSE classification) metrics such as precision or F-measure may be averaged globally over the entire set of classes.

[15] Definition of F-measure from http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score [25/07/2018]

This can be done in two separate ways, after a 2x2 confusion matrix has been prepared for each class separately (Manning & Schütze 1999 p. 577):

- Macro-Averaging: evaluation metrics are computed separately for each confusion matrix, and then averaged over all classes in order to get an overall performance measure. Computing the metric this way gives equal weight to each class, and shows, especially in classification with imbalanced data for important classes, the general quality of the classification results.
- Micro-Averaging: by summing all results from the individual confusion matrices into a single table, a so-called global contingency table (Sebastiani 2002 p. 37) is created. The evaluation metrics are then computed for this table. Computing a metric in this way gives equal weight to each sample, ignoring underlying class imbalance. According to Liu et al. (2007 p. 171), micro-averaging results in more accurate metrics than macro-averaging.

What is most notable about the global contingency table, is the fact that false positives (FP) and false negatives (FN) are always the same. What would be a coincidence for a binary classifier is the logical output of how the global values are calculated. The FP for one class are part of the FN for another class, resulting in identical numbers for FP and FN in the global contingency table. Table 1 shows how to compute the prediction values for an individual confusion matrix, to be repeated over all classes.

		Actual Class				
		electronics	gaming	security	travel	cooking
Predicted Class	electronics	578	24	11	4	1
	gaming	7	518	7	9	6
	security	9	30	576	4	0
	travel	1	17	6	581	6
	cooking	5	11	0	2	587

electronics	TP	FN	FP	TN

Table 1: How to compute the prediction values TP, FN, FP and TN for a single class, using the confusion matrix of a multi-class classifier

Goodfellow et al. (2016 p. 102) note, that it is often difficult to choose a performance metric. This is especially true for multi-class classification problems, as most metrics were "[...] originally developed and applicable for binary classification problems" (Hossin & Sulaiman 2015 p. 7). Most literature is about binary classifiers and how to evaluate such a classifier. In real-world applications though, the data to be classified is not always limited to two classes (ibid.), which makes an informed decision very difficult.

Because of this, the following measurement methods were eventually selected:

- Overall micro-averaged Accuracy, as the training data was perfectly balanced.
- Overall micro-averaged Precision, which is the same as the recall and the F-measure, due to the fact that FP and FN are the same
- Average (arithmetic mean) confidence

The Accuracy and the Precision serve to show the comparison between all correctly classified samples and the fraction of positives that actually are true (TP). The goal is to see the development of the performance over the life of the experiment. As precision is computed over a subset of the classified set, it is represented by a slightly lower number than Accuracy. It also tends to deflect more pronounced on variations. The decision for those two measures was also influenced by the fact that in order to be able to represent classifier performance over time, a single-number metric was required. For this reason, the ROC curve (receiver operating characteristics) was not selected, even though it is a recommended performance measure for classifiers (Fawcett 2006 p. 861; Bakhsh et al. 2018 p. 17). It is a graphical representation which is usually applied to visualize classifier performance for binary classifications problems (Fawcett 2006 p. 871). It is difficult to visualize multiclass classifier performance, as each class needs to be plotted separately. It is therefore only possible to provide a performance measurement for a specific point in time, but not over time.

The average confidence is used to represent the performance and confidence of the classifier at the point of classification. Accuracy and precision are only available if the actual correct label assignments for the classified samples are known. The confidence mean can be computed at the time of the classification process and can serve as an early indicator for concept drift.

6.1 Evaluation of experiment 1

The goal for experiment 1 was to test hypothesis 2, the assumption that the automatic selection of samples from recently classified (labelled) texts which were reviewed by HITL results in a good classification model, with an increase in accuracy compared to previous training instances.

Looking at Figure 19, the hypothesis can neither be validated (in terms of an increased accuracy) nor rejected (in terms of the classification resulting in a good classification model). Both Accuracy and Precision remain stable over the course of the experiment. It is interesting to note though, that if we calculate these same metrics for the low-confidence samples only (i.e. those samples for which the confidence value was below the threshold), their values are significantly lower than that of the overall metrics.

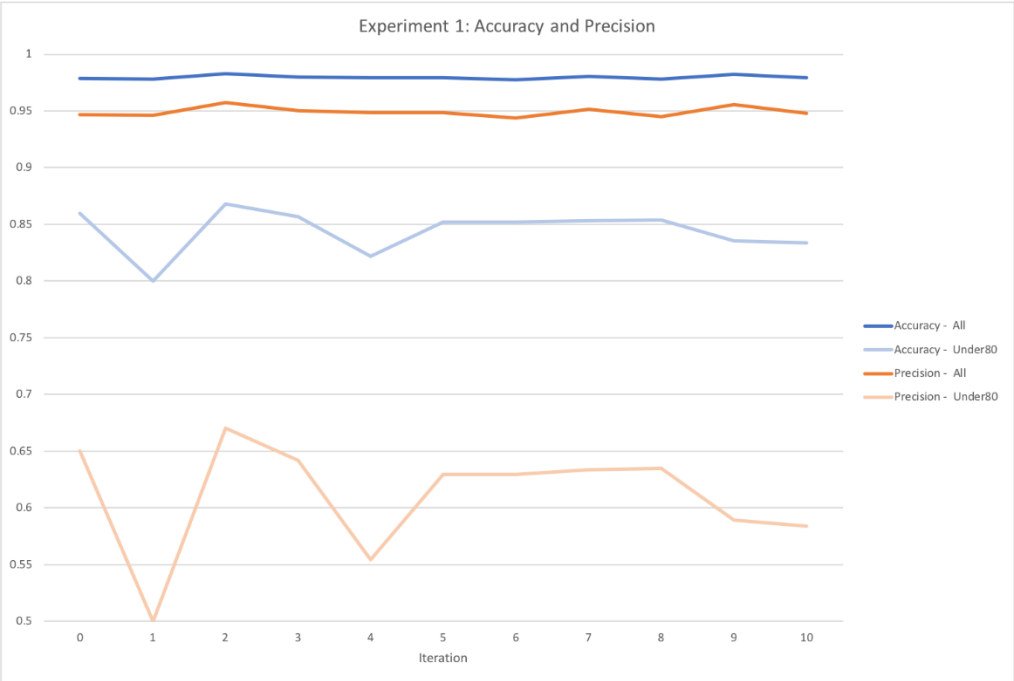

Figure 19: Accuracy and precision values from global contingency table, for the total number of samples as well as for the samples with confidence below threshold for experiment 1.

This is due to the fact that the greater part of the correct classifications is classified with a confidence value (CV) over 80, to be precise 97.287% (77 samples of 2838 were below threshold) for iteration 1. This is represented in the high confidence value mean, showing a confidence mean of 0.964 for iteration 1 (Figure 20). This shows that a high percentage of all label assignments were actually done with a high confidence, to be precise 94.83% (2845 samples of 3000), which leaves 155 label assignments below the CV threshold.

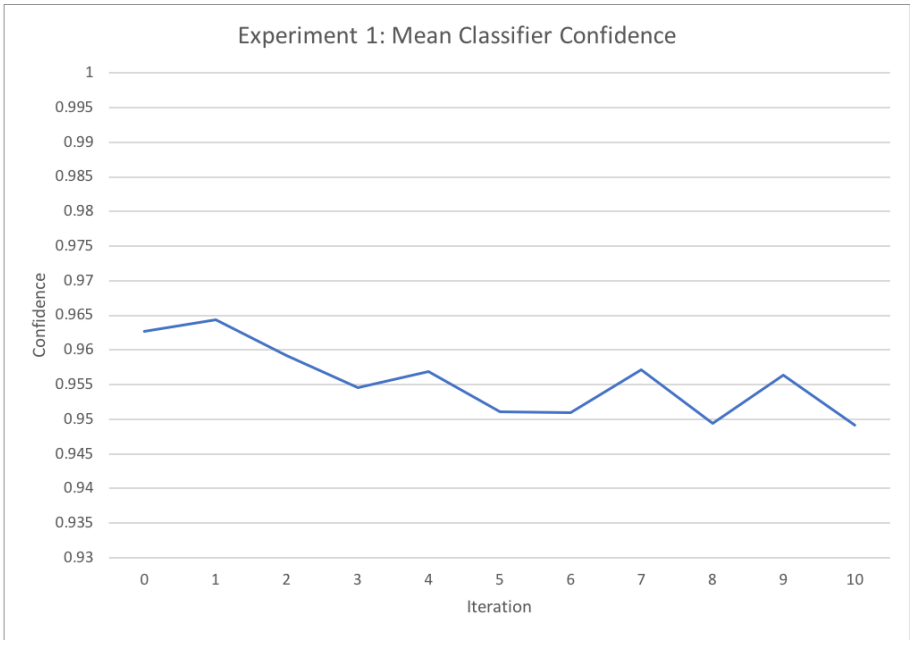

Figure 20: Arithmetic mean of overall confidence value, experiment 1

Coming back to the low precision and accuracy measures for assigned labels below 80%:

In iteration 1, only 79 of those 155 samples below the threshold were labeled correctly (leaving another 76 falsely labeled samples in that tranche), which is reflected in the precision value of 0.5 at that point. This also shows though, that of the 155 FP, 76 will ideally be corrected by HITL resulting in a slightly higher accuracy of 0.9846667 (before HITL intervention: 0.9784)

Hypothesis validation

The approach proposed in this paper therefore results in a higher accuracy for the recent classifier model, whereas no increase in accuracy in the classifier model in the next iteration can be reported.

A quick look at the contingency tables (Table 2) shows the majority of entries in the diagonal, representing correct labels and documenting the overall good classifier accuracy. The display as a heatmap accentuates those class-pairs that are often confused with each other (single cells), or classes that seem to be ill-defined (columns) (Bakis et al. 2017 p. 7). We can see that the class-pair security-electronics is confused with each other fairly often, while the gaming class is suffering from an overall lower prediction accuracy. In a real-world scenario, an SME for this domain should consequently look into this class and check whether it should be split up in different, more narrow classes.

Ex1, It0

Predicted Class \ Actual Class	electronics	gaming	security	travel	cooking
electronics	578	24	11	4	1
gaming	7	518	7	9	6
security	9	30	576	4	0
travel	1	17	6	581	6
cooking	5	11	0	2	587

Ex1, It1

Predicted Class \ Actual Class	electronics	gaming	security	travel	cooking
electronics	571	20	13	2	2
gaming	5	535	11	13	6
security	23	25	568	2	1
travel	0	9	7	578	5
cooking	1	11	1	5	586

Ex1, It5

Predicted Class \ Actual Class	electronics	gaming	security	travel	cooking
electronics	569	17	22	2	2
gaming	3	538	7	5	3
security	20	16	562	3	0
travel	4	20	9	584	3
cooking	4	9	0	6	592

Ex1, It10

Predicted Class \ Actual Class	electronics	gaming	security	travel	cooking
electronics	559	11	10	1	0
gaming	11	529	6	4	1
security	22	23	576	2	0
travel	2	19	8	587	6
cooking	6	18	0	6	593

Table 2: Contingency tables for iterations 0, 1,5 and 10 for experiment 1, over all classes and displayed as heatmap.

6.2 Evaluation of experiment 2

The goal for experiment 2 was to test two hypotheses: H1 which states that concept drift could be detected by analyzing the variation in classifier confidence and H3, which assumes that by applying HITL to detect emerging new classes, concept drift could be countered. As

experiment 2 is based on experiment 1, the findings on the class-pair confusion and the ill-defined class definition apply here as well.

Looking at Figure 21 and Figure 22 both hypotheses can be validated. All of the three metrics *accuracy*, *precision* and *confidence mean* display a noticeable dip in iteration 0a. Looking at the contingency tables in Table 3, (especially It0 and It0a) we notice that concept drift was introduced in iteration 0a, resulting in a high number of falsely classified samples.

Ex2, It0

Predicted Class \ Actual Class	electronics	gaming	security	travel	cooking
electronics	580	17	16	5	2
gaming	3	537	8	8	8
security	15	24	572	2	0
travel	0	19	4	577	9
cooking	2	3	0	8	581

Ex2, It0a

Predicted Class \ Actual Class	electronics	gaming	security	travel	cooking	pets
electronics	574	18	17	6	3	13
gaming	7	536	14	8	10	337
security	12	24	566	6	1	6
travel	4	11	3	574	5	66
cooking	3	11	0	6	581	178
pets	0	0	0	0	0	0

Ex2, It1

Predicted Class \ Actual Class	electronics	gaming	security	travel	cooking	pets
electronics	580	9	9	3	1	1
gaming	5	551	8	2	4	5
security	11	18	577	0	1	0
travel	0	10	5	588	2	3
cooking	0	2	0	5	582	3
pets	4	10	1	2	10	588

Ex2, It10

Predicted Class \ Actual Class	electronics	gaming	security	travel	cooking	pets
electronics	566	16	15	0	1	1
gaming	5	491	3	3	1	0
security	14	24	567	4	0	0
travel	3	26	11	585	8	1
cooking	8	9	0	2	583	7
pets	4	34	4	6	7	591

Table 3: Contingency tables for iterations 0, 0a, 1 and 10 for experiment 2, over all classes and displayed as heatmap. Concept drift introduced with iteration 0a

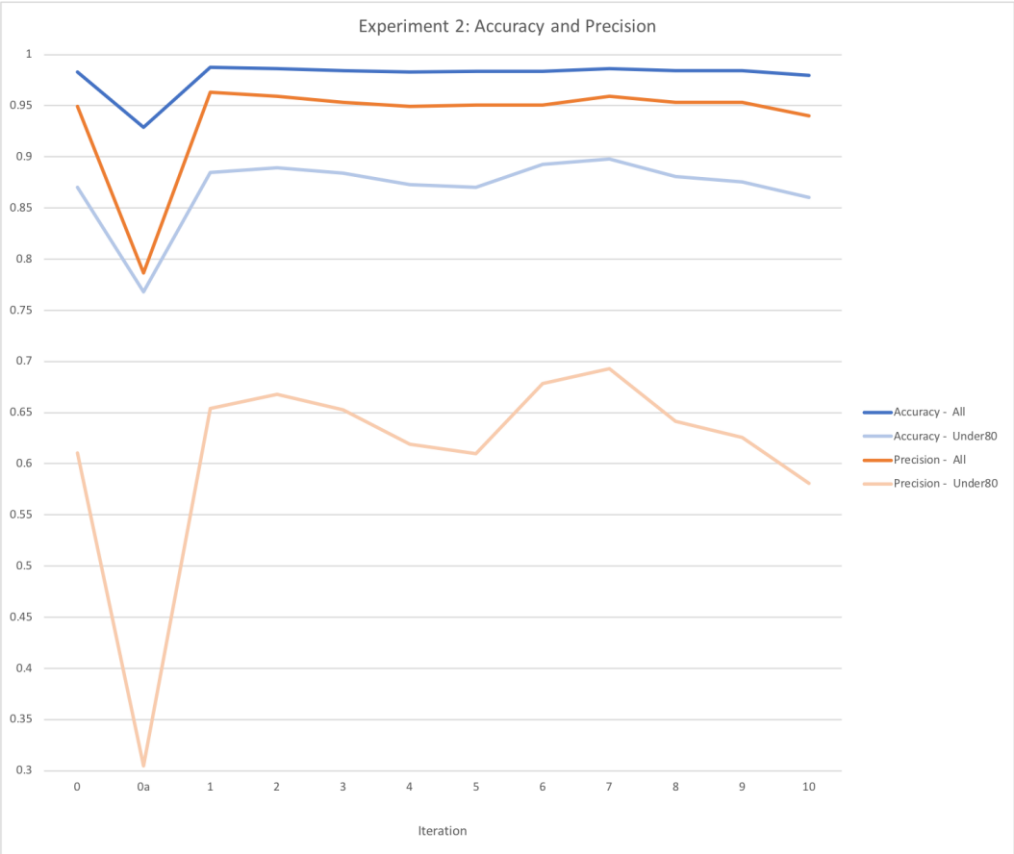

Figure 21: Accuracy and precision values from global contingency table, for the total number of samples as well as for the samples with confidence below threshold for experiment 2.

The dip visible in the average classifier confidence means is due to the high number of low confidence samples for iteration 0a, which have increased by a good third compared to the usual share, resulting in 374 low confidence samples.

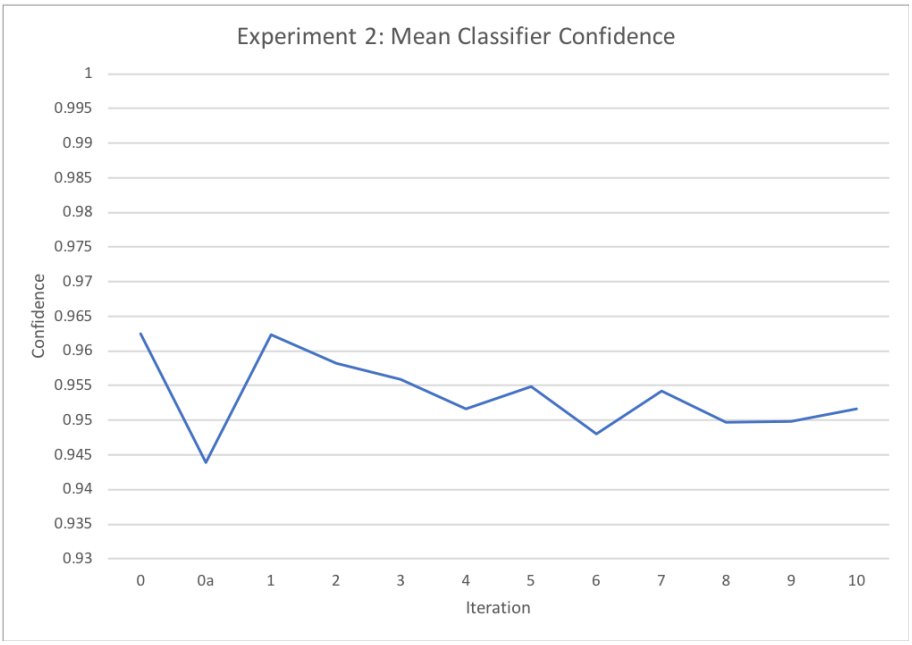

Figure 22: Arithmetic mean of overall confidence value, experiment 2.

Of those 374 samples, 184 samples are from the new class (pets) alone, meaning that 30.667 % of that new emerging class is presented to the HITL for review and correction. This is a fairly high number, and the HITL should be able to recognize the new class. This approach of course is only true for sudden concept drift, where a lot of samples with a new class are coming in at the same time.

Hypothesis validation

The average classifier confidence can be applied as an early indicator for concept drift, depending on the type of concept drift. For a sudden drift, a change should be noticeable. It is necessary though to define the lower bounds on the CV range, to determine which values are still in range, and which are abnormal. The HITL intervention detected the emerging class and formally introduced it to the system by adding it to the training set, thus counteracting concept drift.

6.3 Additional comments on the experiments

In order to be able to get a better idea about the development of the classifier performance, and whether it stays stable, it could prove helpful to add another couple of iterations. While the accuracy in experiment 1 seems to have stabilized at iteration 10, it is difficult to say for Experiment 2.

The use of the Stack Exchange Data approximates a real-world scenario. The Stack Exchange data dump is no data corpus specifically prepared for classification tasks such as the Reuters Corpora[16] (Manning et al. 2009 p. 279) are. As such the Stack Exchange data is prone to misclassifications inherent in the raw data used for the training sets. Slight variations in the accuracy, as are seen in both experiments, could be caused by that.

[16] https://trec.nist.gov/data/reuters/reuters.html

7 Observations and Recommendations: supporting practitioners

The generic architecture presented in chapter 4 should provide a good starting point for a development team tasked with building a classification system. As the author believes sharing the experiences made in the course of developing the solution and the experiments is a good practice, and might help reduce effort and risk during the design and implementation of the proposed system. Some of it will be presented in this chapter. The first part discusses open question and problems which arose during the conception and implementation of the TC system. The second part presents a couple of recommendations to be considered, especially with regard to the classification service or algorithm.

7.1 Open questions and problems

- When implementing the *Re-Train Service*, it should be ensured not only that the training set includes the latest reviewed samples, but also that the maximum number of samples per class is defined, and samples for each class are truncated to that maximum if necessary. This way, it is made sure that the training set will contain enough samples for each and every class that is to be trained. This will prevent the corner case where all the latest reviewed samples are of one single class (possibly because the previous round failed to train for that class correctly), and the samples for that one class thus end up "occupying" all sample spaces in the training set. This would lead to the training of a classifier trained with only one class.
- One question to be considered beforehand, is whether the underlying categorization system needs to be revised. More specifically, whether it is necessary to refile the already categorized documents into new categories, in the case where a new concept would emerge, and documents suddenly might belong to another class. According to Forster (2018) this might depend on the industry, as well as the reason for the emerging concept. In the insurance industry for example, new regulations sometimes require new categorization. Filing old documents according to the new concepts would be wrong though, as the basis for the new categories are new regulations - the old regulations might not be valid anymore, but it is still necessary to be able to connect documents with their respective class definition. Refiling old documents based on a new class system is therefore no option. It depends on the specific use case though, because in the case of emerging subclasses, it might still be necessary to refile the respective documents.
- In addition to the refiling question, there is also the issue whether such classes that are no longer relevant or asked about, should be deleted from the label set. Bakis et al. (2017 p. 9) state that it is an important topic to address how classes can and will

be removed. Because if not, "[...] the number of answer classes will continue to grow and have an impact on accuracy"(ibid.). If there are specific classes that need to stay in the label set, even though they might not occur in the incoming data, this probably needs to be implemented in the Re-Train Service. The respective classes can then be defined in the ruleset, which would allow for a flexible management of the label set.

7.2 Recommendations for development

Invest some time into finding out the "right" sample threshold. The required number of training samples depend greatly on the employed service, respectively the used algorithm(s). Depending on the desired accuracy for the classifier, a learning curve should be computed beforehand. Keep in mind though, that "the more the better" doesn't always hold - the more samples required for a new training cycle, the longer usually the training takes, and the longer it takes for the system to be able to adapt to concept drift. Also, a machine learner most probably will never reach 100% accuracy (Sebastiani 2002 p. 47). A learning curve diagram could be used as a means to find out the minimum and the maximum number of samples - the number of samples needed to achieve the minimum required accuracy, would then be the number of samples required for a new class to be included in the training set, in order to introduce it to the system. More samples will come in after the first iteration with the new class, which can then be added to maximize the number of samples to achieve maximum accuracy. This was not implemented in the PoC but would be feasible.

Carefully select the algorithm(s) or classification service to be used beforehand. One possibility would be to prepare the ground truth with its respective label set and test it on the various possibilities. According to Kitchin (2014 p. 105), it is usually either a Naïve Bayes classifier, a decision tree technique, a neural network or a support vector machine (SVM) used for classification tasks. A ready-made service could also be a good choice, or a combination of methods, called an "ensemble learner". The selection very often depends on the use case and, of course, the skillset available in the organization. All methods have their advantages and drawbacks as well as a specific classification task they usually work well on. A good starting point to read up on text classification algorithms is Aggarwal & Zhai (2012).

Incorporate user feedback where possible and feasible. Classification and correcting samples is a laborious task. The importance of classification through user feedback should not be underestimated. User feedback could be for example the end user moving a document from one folder to another in the external system (e.g. a Document Management System), which would then represent a label correction. Make sure if it is possible to recognize and register such changes as a change of the classification metadata, stored in the storage component of the TC system.

Everybody is a SME in their own field. People responsible for a category of documents represented by a class should be integrated as HITL. One person by itself will not be able to review and correct all classes accordingly.

Examine the impact of new classes on the business environment. It is necessary to consider the extended application landscape beyond the TC system itself regarding what impact defining a new class might have: Identifying and creating new classes in the TC system might entail creating new folders in a file system, a new forum or community in a Knowledge Management system, a new mail filter or a new SharePoint page, to name only a few examples. A new class might have no impact at all, or it could be the cause of subtle issues that are difficult to track down if not treated correctly. It is therefore important to anticipate possible changes to the TC system's environment beforehand, in particular to the external systems directly or indirectly connected to the TC system.

Avoid being restricted by the classifier’s technical limitations. The restriction of the number of characters imposed by IBM NLC was not an issue within this paper, because of the chosen raw data set from Stack Exchange, where the forum post lengths are usually within that character limit. Texts to be classified within a business application are not necessarily that short though, so it is a topic that needs to be addressed. If working with a classifier service such as IBM NLC that does not support long texts, IBM (n.d.-a p. 9) proposes the pre-classification step of de-composition, where long texts are broken into sentences, or paragraphs, for example. These individual text parts would then be individually labeled, and the classification results then aggregated into a kind of summary of the text. A set of well-defined business rules could then be used to trigger actions depending on the composition of the summary. Or a meta-classifier could be applied in order to make a "[...] classification based on the results from the sentence level" (ibid.) labels. Or, chose another algorithm.

8 Conclusion

The goal of this paper was the proposal and evaluation of an architecture for a text classification system that was able to counteract concept drift.

The development of the thesis followed the Design Science Research Methodology Process Model presented in chapter 1.4, which acted as a step-by-step guide. The core principle of DSR is, according to Hevner et al. (2004 p. 82), that knowledge and understanding of a problem and its solution are gained by building an artifact and putting it to application. This was found to be true, and through the application of DSR to the master thesis and by adhering to the DSR guidelines provided by Hevner et al. (2004, p. 83), it was possible to develop a coherent paper.

During the course of this work, four hypotheses were introduced that represent the main underlying assumptions of the proposed system.

Except for hypothesis 2, which postulates that assembling the training set from previously classified and corrected text samples results in an increased classification accuracy, all hypotheses were fully validated:

Hypothesis 2 can be only partially validated, as the selection of reviewed and corrected samples for the training set has neither a positive nor a negative influence on the classifier model accuracy within the following iteration.

Hypothesis 1, assuming that concept drift can be detected through changes in the average classifier confidence, was validated through experiment 2: The classifier confidence can be applied to work as an early indicator for concept drift. Because experiment 1 did not show an improvement in classifier accuracy over the course of the whole experiment, hypothesis 2 couldn't be fully validated, but it was not rejected either. The HITL intervention in combination with the classifier confidence enabled the detection and inclusion of a new class though, which validated Hypothesis 3.

Hypothesis 4 is validated through the system architecture which shows the importance of the metadata stored. Gama et al. (2014 p. 20) also confirms this by stating that retraining approaches as implemented in the proposed system, need some data to be stored separately.

The design of the experiments, in hindsight, may have had one significant flaw: By designing them in a way that samples were drawn from the original corpus of raw data, reality was somewhat oversimplified, and a hidden assumption added that a corpus of previously correctly labeled samples was actually available. The design should have been in a way that all samples were drawn from previously trained batches.

But that design, in turn, would have had its own significant issue, as it necessarily assumes that there are always a sufficient number of recently labelled texts available per class for training, which will not always be the case in a real-world scenario. Hence, in order to be able to prepare a sufficiently large training set for a new model, the person responsible would have to resort to selecting older samples.

Therefore, the two experiments validate the underlying hypotheses 1-3, and by that, the method to detect concept drift using confidence metrics, and reinforcing class labels by using corrected low-confidence samples for re-training. These results, together with the conceptual architecture framework presented in chapter 4, represent in the authors view a successful PoC.

The architecture was developed according to well-known standards and has been validated by two architects independently. To the best of the authors knowledge, no such system has been described or implemented before. The two underlying assumptions for the master thesis, namely that a HITL approach would have a positive influence on accuracy, and that classifier confidence could be used to detect concept drift, have both been described separately and in theory in the literature, but not combined in such a way as proposed in this paper.

The HITL approach is a component of active, interactive and coactive learning, as discussed in chapter 3.3.3, but it is also utilized as a part of the training set creation. IBM (n.d.-a p. 5) calls it a validation classification where a prepared, but not yet labeled training set is classified by the old model, and the outcome is then validated by SMEs. This is an approach similar to the proposed system but is not continuously implemented as it is in this paper.

A field where there seems to be an increasing number of researches towards a human-in-the-loop approach, often referred to as an expert-in-the-loop, are the medical domains (Holzinger 2016; Yimam, Biemann, Majnaric, Šabanović, & Holzinger 2016), where the expert knowledge of an SME is of high relevance, due to the high cost occurring in case of a misclassification.

Using the classifier confidence as a method to detect concept drift has been described as an active learning strategy (Žliobaitė et al. 2014 p. 37 and 41), and can be seen in applications where a user is asked for feedback in cases where the system is not confident enough. This can be seen in e.g. photo applications where face recognition is enabled. After an initial training phase, the frequency with which the application asks the user for feedback steadily decreases. If the application does not recognize a face, experiencing low confidence, it will ask for feedback again, such as whether the person automatically recognized in a picture is indeed this person. These are applications using a continuously learning algorithm though,

while the proposed system in this paper is employing a batch-learning service where this is not possible.

Not all businesses that need a classification system can or want to design and build such a machine learner by themselves, and the proposed system offers a feasible approach to circumvent static classification without the need for exhaustive machine learning knowledge.

Manning et al. (2009 p. 338) briefly describes a similar approach as a hybrid solution, where manual and automatic classification were combined, and noted that this might be the required step to achieve stable accuracy over time. Such an approach produces new training data for future classifier models, as described in chapters 3.1.2 and 3.1.3. They conclude that this would be a case where training data for a new model is "[...] clearly not randomly sampled from the space of documents." (ibid.) But perhaps the stipulation that training data always need to be randomly sampled is an imperative that has outlived its usefulness for real-world applications.

The system architecture presented in chapter 4 should provide a good starting point for a practitioner (such as an IT architect) with building an effective text classification application. It is important though to keep in mind that the "[...] real-world deployment of techniques that have proven successful in the laboratory often meet with challenging practical problems." (Forman & George 2006 p. 252) Although the experiments tried to model a development over time, it was not possible to model potential variations in the text, or a data stream.

8.1 Limitations

A paper such as this, where experiments are incorporated in order to validate an assumption, always suffers from the problem that a real-world scenario is not easy to replicate in an experiment, and it would often not be sensible either. It is necessary to check whether a method works in principle and not which influences have to be considered. This would then probably be done in a subsequent step. It is therefore easier to demonstrate a method or how something works in a well-defined narrow setting, as is usually the case within an experiment. This is also the case with text classification, where most classification algorithms or services work well on small label sets. As Manning et al. (2009 p. 337) state though, real classification problems often consist of large numbers of frequently similar categories. To achieve accurate classification over such large label sets is difficult. Even though the experiments were conducted within a multi-class setting, 5 respectively 6 classes for experiment 2, is still a small label set.

As seen in chapter 2.2.2, concept drift can occur in various ways. This master thesis addresses only one type of concept drift: sudden true concept drift, specifically the novel class appearance. The validation for other types of drift is therefore missing. How the

proposed system would react to other types of concept drift could not be researched in this paper. It thus represents a limitation, but also a future research question.

The combination of the HITL approach with the average classifier confidence as indicator, enables the system to recognize new classes. It is not possible though to recognize the emergence of subclasses. This could e.g. be happening in the Gaming category (see chapter 5.3 Experiment 2): all posts related to games are categorized in there, but the emergence of a new separate game sub-board, for example for a new game like Fallout 4 could not be detected. This would need to be done manually. As we have seen in chapter 6.2, Table 3 the confusion matrix could be indicating just this. That being said, such a confusion matrix would not be available in a productive setting, as it can only be computed if knowledge about the true class labels is available. This would typically not be the case in the productive phase, only in the training and testing phase.

The HITL aspect could only be tested in the experiment in the form of corrective measures on the samples identified for review. This method always assumes the ideal case in which the HITL always assigns the correct labels without fail. This is in reality never so, as Sebastiani (2002 p. 47) pointed out. The HITL aspect should therefore also be tested for validation. The development and application of the HITL interfaces in a real-world scenario, even if small, would be advisable, in order to get feedback on the best HITL methods (re-classify, evaluate and set productive).

8.2 Future research and possible next steps

Based on the experiments conducted during the course of this paper, future work in the area of detecting concept drift and re-training classifiers within the context of a business application could be focused on the following points:

Reasons for retraining a classifier: At least two reasons for retraining a classifier in such a setting can be identified. One would be reaching the sample threshold of a new class, as is tested in experiment 2, and which lies within the scope of this paper. The other would be the case when the classifier accuracy is falling below a certain value. The accuracy is bound to decline over time, the longer the classifier model is used. This is due to the fact that in the beginning the accuracy is a measurement of the quality of the classifier in relation to a modeled view of the environment at that point in time. But, as we have seen, the environment and thus the data originating from it change over time. Automatic classifications therefore typically need to be manually adjusted to these changes. This results in a constantly dropping accuracy rate over time. At some point, the least acceptable accuracy level is reached, where system users probably have to revise more classifications than are classified correctly. This should then trigger re-classification efforts. The question here is twofold:

a) what would be the least acceptable accuracy before re-training needs to take place, and

b) how can this actually be measured.

Incremental or gradual concept drift also needs to be counteracted: What could be a good indicator for these types of concept drift?

Detection of emerging subclasses: As discussed in above, subclasses of existing classes are not that easy to detect. What kind of approach would be needed to facilitate either the detection of subclasses, or the nature of a new class, its content and context?

Composition of the training set: Will there be a notable difference in accuracy if a new model is to be trained just from random samples from the newly classified batch instead of the reviewed samples, plus random samples from the classified batch? The experiments implemented for this paper use a randomly selected sample from the dataset, where all classifications are correct. Corrected low-confidence samples are then added, for which the corrected value is always accurate. Real-world scenarios differ on both counts, raising additional questions.

Firstly, the training set samples would necessarily be selected from previously classified texts. This classification is done by the classifier itself, which has an error rate that is non-zero. This means that a non-zero percentage of the training data is inaccurate. The question then is, what the decrease in accuracy due to this inaccurate training data would be. This decrease in accuracy is compensated by manual corrections of the classification, but it is unreasonable to expect this manual correction to be done on all misclassified samples, because not all samples can be reviewed, since some have label assignments with a high confidence value. Some assumption on which classifications to review need to be made: otherwise, all classification will have to be reviewed at all times, which is neither feasible nor sensible. All misclassified samples with a high confidence value stay therefore undetected. Another question would then be, what amount of corrections would be necessary to counteract this loss of accuracy.

Secondly, the accuracy of manual classifications is not absolute (Sebastiani 2002 p. 47). This of course means that the correction of the misclassified samples will not result in 100% correct labels. The last question therefore would be, how the accuracy of this manual classification would affect the final accuracy of the classifier.

These are just some research questions to encourage researchers to continue exploring new methods and approaches that will support practitioners in improving the performance of text classification systems in business applications.

9 Bibliography

Aggarwal, Charu C.; Zhai, ChengXiang. (2012): A Survey of Text Classification Algorithms. In *Mining Text Data* (pp. 163–222). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4614-3223-4_6

Amershi, Saleema; Cakmak, Maya; Knox, William Bradley; Kulesza, Todd. (2014): Power to the People: The Role of Humans in Interactive Machine Learning. *AI Magazine*, *35*(4), 105. https://doi.org/10.1609/aimag.v35i4.2513

Auberson, Frederic; Auberson, Kirsten Scherer. (2018): Experiments - Code. Retrieved from https://github.com/KSAub/MT2018_PoC_ConceptDrift

Bakhsh, Iqbal; Mohammad, Zia; Berajawala, Suneil. (2018): *Natural Language Classifier*.

Bakis, R.; Connors, D. P.; Dube, P.; Kapanipathi, P.; Kumar, A.; Malioutov, D.; Venkatramani, C. (2017): Performance of natural language classifiers in a question-answering system. *IBM Journal of Research and Development*, *61*(4), 14:1-14:10. https://doi.org/10.1147/JRD.2017.2711719

Brodley, Carla E.; Friedl, Mark A. (1999): Identifying Mislabeled Training Data. *Journal of Artificial Intelligence Research*, *11*, 131–167. https://doi.org/10.1613/jair.606

Bryman, Alan; Bell, Emma. (2015): *Business Research Methods* (4th ed.). Oxford University Press. https://doi.org/0195430298

Carrillo, Henry; Brodersen, Kay H.; Castellanos, José A. (2014): Probabilistic performance evaluation for multiclass classification using the posterior balanced accuracy. *Advances in Intelligent Systems and Computing*, *252*, 347–361. https://doi.org/10.1007/978-3-319-03413-3_25

Cook, Denise; Cripps, Peter J.; Spaas, Philippe. (2008): An introduction to the IBM Views and Viewpoints Framework for IT systems. IBM developerWorks. Retrieved from https://www.ibm.com/developerworks/rational/library/08/0108_cooks-cripps-spaas/

Fawcett, Tom. (2006): An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Feldman, Ronen; Sanger, James. (2007): *The text mining handbook : advanced approaches in analyzing unstructured data*. Cambridge University Press.

Ferreira, Maria João. (2017): *Workflow Recommendation for Text Classification Problems*. Universidade do Porto. Retrieved from https://repositorio-aberto.up.pt/bitstream/10216/107762/2/219435.pdf

Forman, George; George. (2006): Tackling concept drift by temporal inductive transfer. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '06* (p. 252). New York, New York, USA: ACM Press. https://doi.org/10.1145/1148170.1148216

Forster, Johannes (IBM). (2018, May 17): In-Depth Interview.

Frauchiger, Daniel. (2017): Anwendungen von Design Science Research in der Praxis. In *Wirtschaftsinformatik in Theorie und Praxis* (pp. 107–118). Wiesbaden: Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-17613-6_8

Gama, João; Žliobaitė, Indrė; Bifet, Albert; Pechenizkiy, Mykola; Bouchachia, Abdelhamid. (2014): A survey on concept drift adaptation. *ACM Computing Surveys*, *46*(4), 1–37. https://doi.org/10.1145/2523813

Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron. (2016): *Deep learning*. MIT Press. Retrieved from http://www.deeplearningbook.org/

Gurevych, Iryna; Meyer, Christian M.; Binnig, Carsten; Fürnkranz, Johannes; Kersting, Kristian; Roth, Stefan; Simpson, Edwin. (2017): *Interactive Data Analytics for the Humanities*. Retrieved from http://www.informatik.tu-darmstadt.de

Hand, David J. (2006): Classifier Technology and the Illusion of Progress. *Statistical Science*, *21*(1), 1–15. https://doi.org/10.1214/088342306000000060

Hartson, H. Rex.; Pyla, Pardha S. (2012): *The UX Book : process and guidelines for ensuring a quality user experience*. Morgan Kaufmann/Elsevier. Retrieved from https://www.safaribooksonline.com/library/view/the-ux-book/9780123852410/

Henrich, Andreas. (2008): *Information Retrieval 1*.https://doi.org/10.1007/s10791-008-9059-7

Hevner, Alan R.; March, Salvatore T.; Park, Jinsoo; Ram, Sudha. (2004): Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75. https://doi.org/10.2307/25148625

Holzinger, Andreas. (2016): Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, *3*(2), 119–131. https://doi.org/10.1007/s40708-016-0042-6

Hossin, M.; Sulaiman, M. N. (2015): Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, *5*(2), 1–11. https://doi.org/10.5121/ijdkp.2015.5201

Hurwitz, Judith; Kaufman, Marcia; Bowles, Adrian. (2015): *Cognitive computing and big data analytics*. Wiley. Retrieved from https://www.wiley.com/en-us/Cognitive+Computing+and+Big+Data+Analytics-p-9781118896624

IBM. (n.d.-a). *IBM Watson Natural Language Classifier: Links, Best Practices, & Design Patterns*. Retrieved from https://stackoverflow.com/questions/tagged/ibm-watson-cognitive

IBM. (n.d.-b). IBM Watson Natural Language Classifier. Retrieved April 30, 2018, from https://console.bluemix.net/docs/services/natural-language-classifier/getting-started.html#natural-language-classifier

IBM. (2016): Artifact: Technical Proof of Concept. Retrieved August 4, 2018, from http://method.ibm.com/rmchtml_soma_ad/index.htm#core.tech.common.extend-ibm_lic/workproducts/technical_proof_of_concept_CEDA4198.html

IBM. (2018a):. Architectural Thinking - Functional Aspect. IBM.

IBM. (2018b):. Architectural Thinking - Requirements Aspect. IBM.

Jurafsky, Daniel; Martin, James H. (2008): *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. *Pearson*. Prentice Hall. Retrieved from https://dl-acm-org.ezproxy.fh-htwchur.ch/citation.cfm?id=555733&preflayout=flat

Kao, Anne; Poteet, Stephen R. (Eds.). (2007): *Natural language processing and text mining*. London: Springer London. https://doi.org/10.1007/978-1-84628-754-1

Kitchin, Rob. (2014): *The Data Revolution: Big Data, Open Data, Data Infrastructures & Their Consequences*. 1 Oliver's Yard, 55 City Road, London EC1Y 1SP United Kingdom : SAGE Publications Ltd. https://doi.org/10.4135/9781473909472

Kuechler, Bill; Vaishnavi, Vijay. (2008): On theory development in design science research: anatomy of a research project. *European Journal of Information Systems*, *17*(5), 489–504. https://doi.org/10.1057/ejis.2008.40

Leek, Jeff. (2015): *The Elements of Data Analytic Style*. *Leanpub*. Retrieved from https://leanpub.com/datastyle/

Liu, Ying; Loh, Han Tong; Youcef-Toumi, Kamal; Tor, Shu Beng Tor. (2007): Handling of Imbalanced Data in Text Classification: Category-Based Term Weights. In A. Kao & S. R. Poteet (Eds.), *Natural Language Processing and Text Mining* (pp. 171–192).

Manning, Christopher D.; Ragahvan, Prabhakar; Schütze, Hinrich. (2009): *An Introduction to Information Retrieval*. *Information Retrieval*. Cambridge University Press. https://doi.org/10.1109/LPT.2009.2020494

Manning, Christopher D.; Schütze, Hinrich. (1999): *Foundations of Statistical Natural Language Processing*. Cambridge: MIT Press. Retrieved from http://ics.upjs.sk/~pero/web/documents/pillar/Manning_Schuetze_StatisticalNLP.pdf

Marr, Bernard. (2018): How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read. Retrieved July 25, 2018, from https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#15ae709d60ba

Munir, Sirajum; Stankovic, John a.; Liang, Chieh-Jan Mike; Lin, Shan. (2013): Cyber Physical System Challenges for Human-in-the-Loop Control. *The 8th International Workshop on Feedback Computing*.

Nunes, D. S.; Zhang, P.; Silva, J. Sá. (2015): A Survey on Human-in-the-Loop Applications Towards an Internet of All. *IEEE Communications Surveys & Tutorials*, *17*(2), 944–965. https://doi.org/10.1109/COMST.2015.2398816

Offermann, Philipp; Levina, Olga; Schönherr, Marten; Bub, Udo. (2009): Outline of a design science research process. *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology - DESRIST '09*, (January), 1. https://doi.org/10.1145/1555619.1555629

Österle, H.; Becker, J.; Frank, U.; Hess, T.; Karagiannis, D.; Krcmar, H.; Sinz, E. J. (2010): Richtungsdiskussionen in der Wirtschaftsinformatik. *Zfbf*, *62*(September), 662–672. https://doi.org/10.1057/ejis.2010.55

Peffers, Ken; Tuunanen, Tuure; Rothenberger, Marcus A.; Chatterjee, Samir. (2007): A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Peng, Roger D.; Matsui, Elizabeth. (2015): *The Art of Data Science: A Guide for Anyone Who Works with Data*. Retrieved from http://leanpub.com/artofdatascience

POWERS, D. M. W. (2011): Evaluation: From Precision, Recall and F-Measure To Roc, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*, *2*(1), 37–63. https://doi.org/10.1.1.214.9232

Rat für Forschung und Technologieentwicklung. (2013): *Empfehlung zu einer optimierten Proof-of-Concept-Unterstützung im Wissenstransfer* (Vol. 43). Retrieved from http://www.rat-fte.at/tl_files/uploads/Empfehlungen/131203_ProofOfConcept_Empfehlung_NP.pdf

Russell, Stuart J.; Norvig, Peter. (2016): *Artificial intelligence : a modern approach* (Third ed.). Boston: Pearson Prentice Hall.

Schirner, G.; Erdogmus, D.; Chowdhury, K.; Padir, T. (2013): The Future of Human-in-the-Loop Cyber-Physical Systems. *Computer*, *46*(1), 36–45. https://doi.org/10.1109/MC.2013.31

Sebastiani, Fabrizio. (2002): Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1–47. https://doi.org/10.1145/505282.505283

Shivaswamy, Pannaga; Joachims, Thorsten. (2015): Coactive Learning. *Journal of Artificial Intelligence Research*, *53*, 1–40. https://doi.org/10.1613/JAIR.4539

Stanton, Jeffrey. (2013): Introduction to Data Science, 196. https://doi.org/10.1007/978-3-319-20424-6

Stock, Wolfgang G.; Stock, Mechtild. (2013): *Handbook of Information Science*. Berlin, Boston: DE GRUYTER SAUR. https://doi.org/10.1515/9783110235005

Strauch, Dietmar. (2004): Grundlagen der praktischen Information und Dokumentation. Bd. 2 Glossar. In R. Kuhlen, T. Seeger, & D. Strauch (Eds.) (5th ed.). Saur.

Sun, Yu; Tang, Ke; Zhu, Zexuan; Yao, Xin. (2018): Concept Drift Adaptation by Exploiting Historical Knowledge. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2017.2775225

Tsymbal, Alexey. (2004): The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, *4*(C), 2004–15. https://doi.org/10.1.1.58.9085

van Aken, Joan; Chandrasekaran, Aravind; Halman, Joop. (2016): Conducting and publishing design science research: Inaugural essay of the design science department of the Journal of Operations Management. *Journal of Operations Management*, *47–48*, 1–8. https://doi.org/10.1016/J.JOM.2016.06.004

Webb, Geoffrey I.; Hyde, Roy; Cao, Hong; Nguyen, Hai Long; Petitjean, Francois. (2016): Characterizing concept drift. *Data Mining and Knowledge Discovery*, *30*(4). https://doi.org/10.1007/s10618-015-0448-4

Widmer, Gerhard; Kubat, Miroslav. (1993): Effective learning in dynamic environments by explicit context tracking (pp. 227–243). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-56602-3_139

Widmer, Gerhard; Kubat, Miroslav. (1996): Learning in the presence of concept drift and hidden contexts. *Machine Learning*, *23*(1), 69–101. https://doi.org/10.1007/BF00116900

Yimam, Seid Muhie; Biemann, Chris; Majnaric, Ljiljana; Šabanović, Šefket; Holzinger, Andreas. (2016): An adaptive annotation approach for biomedical entity and relation recognition. *Brain Informatics*, *3*(3), 157–168. https://doi.org/10.1007/s40708-016-0036-4

Žliobaitė, Indrė. (2010): Learning under Concept Drift: an Overview. Retrieved from http://arxiv.org/abs/1010.4784

Žliobaitė, Indrė; Bifet, Albert; Pfahringer, Bernhard; Holmes, Geoffrey. (2014): Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(1), 27–39. https://doi.org/10.1109/TNNLS.2012.2236570

Žliobaitė, Indrė; Pechenizkiy, Mykola; Gama, João. (2016): An Overview of Concept Drift Applications (pp. 91–114). Springer, Cham. https://doi.org/10.1007/978-3-319-26989-4_4

10 Appendix: Information Systems Research Framework

Information

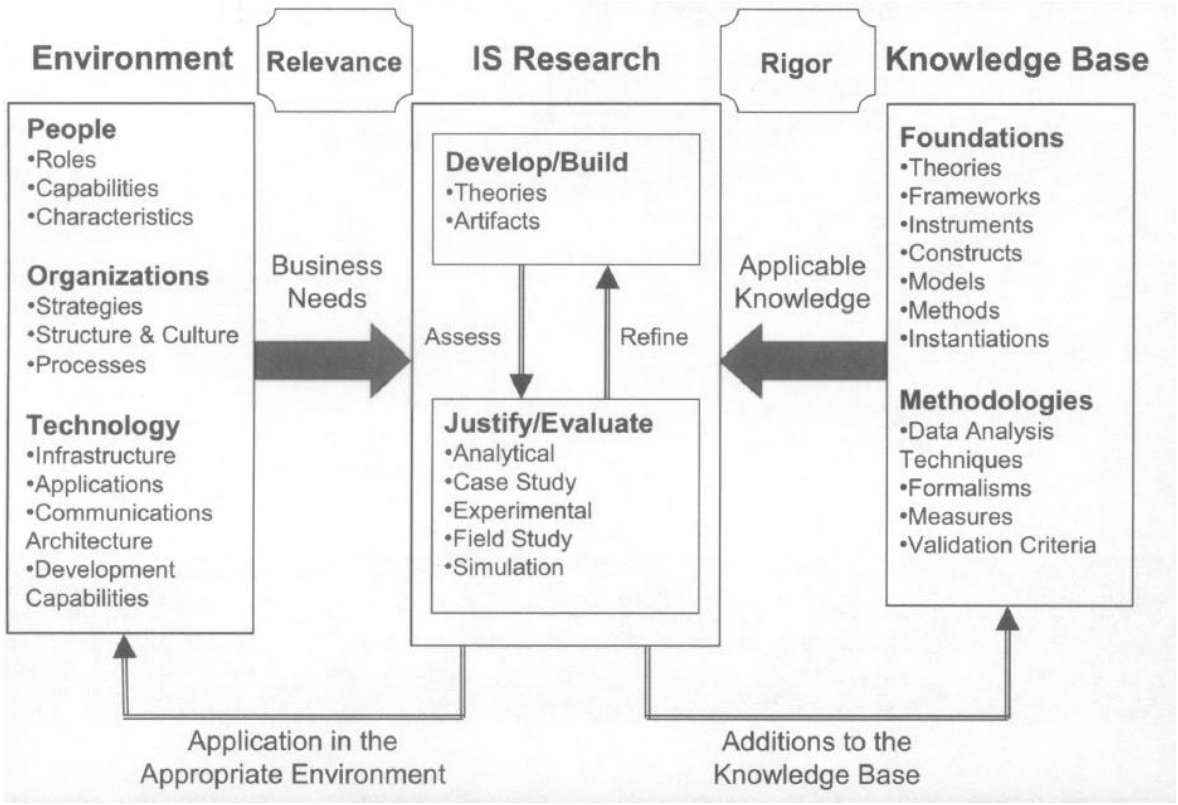

Figure 23: Information Systems Research Framework (Hevner et al. 2004 p. 80)

11 Appendix: Mail exchange with IBM NLC Product Offering Manager

Tue, Mar 13

Iqbal Bakhsh to me

Hi Kirsten,

Confidence scores of NLC are relative i.e. the total scores sums up to 1. Furthermore, confidence levels of classes really depends on training data. As you can imagine, if you have higher quality training data that is a good representative of your actual data, then you will see higher confidence levels.

While NLC does provide confidence scores, we do not expose our underlying parameters. I believe you can make your prototype on the confidence scores that NLC provides. We may not be able to expose our algorithms and their computation.

Best,

Iqbal Bakhsh

----- Original message -----
From: Kirsten Scherer Auberson/Switzerland/IBM
To: Iqbal Bakhsh/New York/IBM@IBMUS
Cc:
Subject: Watson NLC - question about the confidence value
Date: Mon, Mar 12, 2018 12:04 PM

Hi Iqbal,

I'm contacting you because I need information on the confidence value of the Natural Language Classifier service, specifically on what basis it is computed.

I am currently working on my master thesis (for IBM, I'm a Master@IBM student), which is the design and proof of concept for an natural language classification system that will "autonomously" retrain itself, but with human intervention (human-in-the-loop). My plan is to build a small prototype with a Natural Language Classifier-System (preferably Watson NLC). This means of course, that the confidence value is key - once it's below a certain threshold, the human-in-the-loop part is triggered.

As the confidence value is this important, I will need more information on how the confidence is computed. Unfortunately, there is no background information available in the documentation.

Would you be able to provide me with in-depth information on the confidence value? Or forward my mail to someone who can?

I would very much appreciate if you could help me here. Thank you in advance for your help.

Mit freundlichem Gruss / best regards

Kirsten Scherer Auberson

Cognitive Solutions Industry

Master@IBM

IBM Switzerland

---

12 Appendix Experiment 1: graphical representation

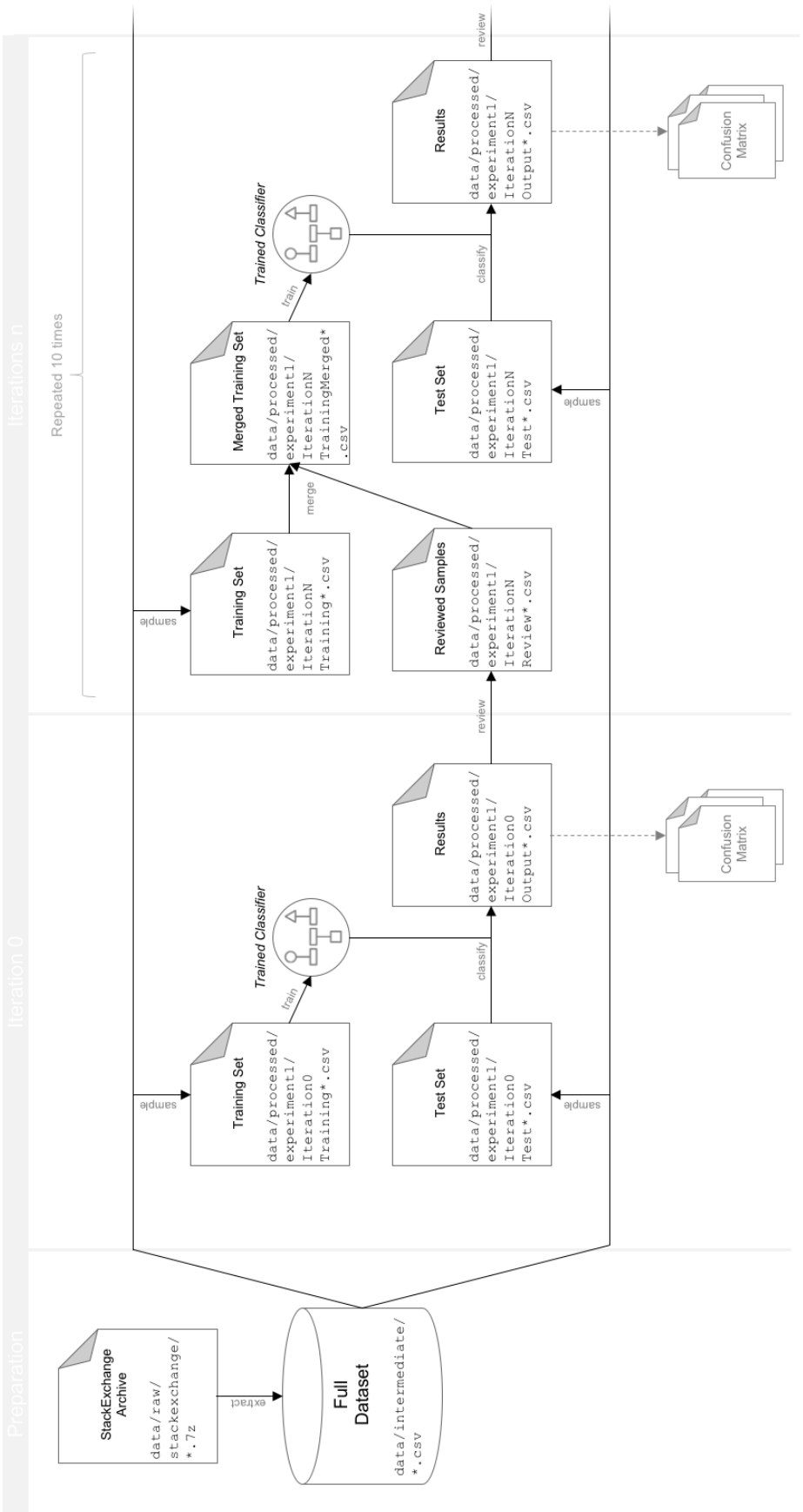

Figure 24: Graphical representation of Experiment1(own depiction)

13 Appendix Experiment 2: graphical representation

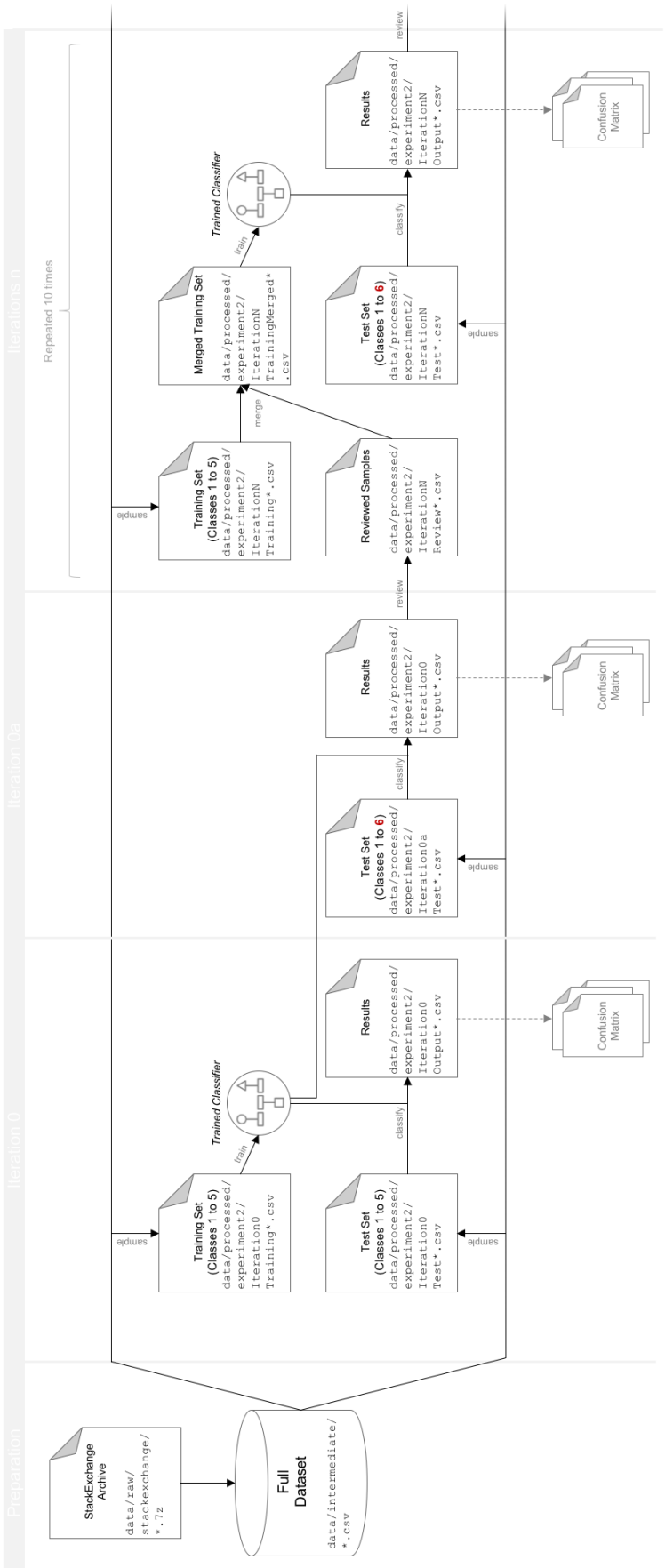

Figure 25: Graphical representation of Experiment 2 (own depiction)

Bisher erschienene Schriften

Ergebnisse von Forschungsprojekten erscheinen jeweils in Form von Arbeitsberichten in Reihen. Sonstige Publikationen erscheinen in Form von alleinstehenden Schriften.

Derzeit gibt es in den Churer Schriften zur Informationswissenschaft folgende Reihen:
Reihe Berufsmarktforschung

Churer Schriften zur Informationswissenschaft – Schrift 1
Herausgegeben von Josef Herget und Sonja Hierl
Reihe Berufsmarktforschung – Arbeitsbericht 1:
Josef Herget
Thomas Seeger
Zum Stand der Berufsmarktforschung in der Informationswissenschaft in deutschsprachigen Ländern
Chur, 2007 (im Druck)
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 2
Herausgegeben von Josef Herget und Sonja Hierl
Reihe Berufsmarktforschung – Arbeitsbericht 2:
Josef Herget
Norbert Lang
Berufsmarktforschung in Archiv, Bibliothek, Dokumentation und in der Informationswirtschaft:
Methodisches Konzept
Chur, 2007 (im Druck)
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 3
Herausgegeben von Josef Herget und Sonja Hierl
Reihe Berufsmarktforschung – Arbeitsbericht 3:
Josef Herget
Norbert Lang
Gegenwärtige und zukünftige Arbeitsfelder für Informationsspezialisten in privatwirtschaftlichen Unternehmen und öffentlich-rechtlichen Institutionen
Chur, 2004
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 4
Herausgegeben von Josef Herget und Sonja Hierl
Sonja Hierl
Die Eignung des Einsatzes von Topic Maps für e-Learning
Vorgehensmodell und Konzeption einer e-Learning-Einheit unter Verwendung von Topic Maps
Chur, 2005
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 5
Herausgegeben von Josef Herget und Sonja Hierl
Nina Braschler
Realisierungsmöglichkeiten einer Zertifizierungsstelle für digitale Zertifikate in der Schweiz
Chur, 2005
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 6
Herausgegeben von Josef Herget und Sonja Hierl
Reihe Berufsmarktforschung – Arbeitsbericht 4:
Ivo Macek
Urs Naegeli
Postgraduiertenausbildung in der Informationswissenschaft in der Schweiz:
Konzept – Evaluation – Perspektiven
Chur, 2005
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 7
Herausgegeben von Josef Herget und Sonja Hierl
Caroline Ruosch
Die Fraktale Bibliothek:
Diskussion und Umsetzung des Konzepts in der deutschsprachigen Schweiz.
Chur, 2005
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 8
Herausgegeben von Josef Herget und Sonja Hierl
Esther Bättig
Information Literacy an Hochschulen
Entwicklungen in den USA, in Deutschland und der Schweiz
Chur, 2005
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 9
Herausgegeben von Josef Herget und Sonja Hierl
Franziska Höfliger
Konzept zur Schaffung einer Integrationsbibliothek in der Pestalozzi-Bibliothek Zürich
Chur, 2005
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 10
Herausgegeben von Josef Herget und Sonja Hierl
Myriam Kamphues
Geoinformationen der Schweiz im Internet:
Beurteilung von Benutzeroberflächen und Abfrageoptionen für Endnutzer
Chur, 2006
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 11
Herausgegeben von Josef Herget und Sonja Hierl
Luigi Ciullo
Stand von Records Management in der chemisch-pharmazeutischen Branche
Chur, 2006
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 12
Herausgegeben von Josef Herget und Sonja Hierl
Martin Braschler, Josef Herget, Joachim Pfister, Peter Schäuble, Markus Steinbach, Jürg Stuker
Evaluation der Suchfunktion von Schweizer Unternehmens-Websites
Chur, 2006
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 13
Herausgegeben von Josef Herget und Sonja Hierl
Adina Lieske
Bibliotheksspezifische Marketingstrategien zur Gewinnung von Nutzergruppen:
Die Winterthurer Bibliotheken
Chur, 2007
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 14
Herausgegeben von Josef Herget und Sonja Hierl
Christina Bieber, Josef Herget
Stand der Digitalisierung im Museumsbereich in der Schweiz
Internationale Referenzprojekte und Handlungsempfehlungen
Chur, 2007
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 15
Herausgegeben von Josef Herget und Sonja Hierl
Sabina Löhrer
Kataloganreicherung in Hochschulbibliotheken
State of the Art Überblick und Aussichten für die Schweiz
Chur, 2007
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 16
Herausgegeben von Josef Herget und Sonja Hierl
Heidi Stieger
Fachblogs von und für BibliothekarInnen – Nutzen, Tendenzen
Mit Fokus auf den deutschsprachigen Raum
Chur, 2007
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 17
Herausgegeben von Josef Herget und Sonja Hierl
Nadja Kehl
Aggregation und visuelle Aufbereitung von Unternehmensstrategien mithilfe von Recherche-Codes
Chur, 2007
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 18
Herausgegeben von Josef Herget und Sonja Hierl
Rafaela Pichler
Annäherung an die Bildsprache – Ontologien als Hilfsmittel für Bilderschliessung und Bildrecherche in Kunstbilddatenbanken
Chur, 2007
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 19
Herausgegeben von Josef Herget und Sonja Hierl
Jürgen Büchel
Identifikation von Marktnischen – Die Eignung verschiedener Informationsquellen zur Auffindung von Marktnischen
Chur, 2007
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 20
Herausgegeben von Josef Herget und Sonja Hierl
Andreas Eisenring
Trends im Bereich der Bibliothekssoftware
Chur, 2007
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 21
Herausgegeben von Josef Herget und Sonja Hierl
Lilian Brändli
Gesucht – gefunden? Optimierung der Informationssuche von Studierenden in wissenschaftlichen Bibliotheken
Chur, 2007
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 22
Herausgegeben von Josef Herget und Sonja Hierl
Beatrice Bürgi
Open Access an Schweizer Hochschulen – Ein praxisorientierter Massnahmenkatalog für Hochschulbibliotheken zur Planung und Errichtung von Institutional Repositories
Chur, 2007
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 23
Herausgegeben von Josef Herget und Sonja Hierl
Darja Dimitrijewitsch, Cécile Schneeberger
Optimierung der Usability des Webauftritts der Stadt- und Universitätsbibliothek Bern
Chur, 2007
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 24
Herausgegeben von Nadja Böller, Josef Herget und Sonja Hierl
Brigitte Brüderlin
Stakeholder-Beziehungen als Basis einer Angebotsoptimierung
Chur, 2008
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 25
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Hans-Dieter Zimmermann
Jonas Rebmann
Web 2.0 im Tourismus, Soziale Webanwendungen im Bereich der Destinationen
Chur, 2008
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 26
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Hans-Dieter Zimmermann
Isabelle Walther
Idea Stores, ein erfolgreiches Bibliothekskonzept aus England – auf für die Schweiz?
Chur, 2008
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 27
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Hans-Dieter Zimmermann
Scherer Auberson Kirsten
Evaluation von Informationskompetenz: Lässt sich ein Informationskompetenzzuwachs messen?
Eine systematische Evaluation von Messverfahren
Chur, 2009 (im Druck)
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 28
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Hans-Dieter Zimmermann
Nadine Wallaschek
Datensicherung in Bibliotheksverbünden.
Empfehlungen für die Entwicklung von Sicherheits- und Datensicherungskonzepten in Bibliotheksverbünden
Chur, 2009
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 29
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Hans-Dieter Zimmermann
Laura Tobler
Recherchestrategien im Internet
Systematische Vorgehensweisen bei der Suche im Internet, dargestellt anhand ausgewählter Fallstudien
Chur, 2009
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 30
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Hans-Dieter Zimmermann
Bibliotheken und Dokumentationszentren als Unternehmen:
Antworten von Bibliotheken und Dokumentationszentren auf die Herausforderungen der digitalen Gesellschaft
Chur, 2009
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 31
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Hans-Dieter Zimmermann
Karin Garbely, Marita Kieser
Mystery Shopping als Bewertungsmethode der Dienstleistungsqualität von wissenschaftlichen Bibliotheken
Chur, 2009
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 32
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Hans-Dieter Zimmermann
Tristan Triponez
E-Mail Records Management
Die Aufbewahrung von E-Mails in Schweizer Organisationen als technische, rechtliche und organisatorische Herausforderung
Chur, 2009
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 33
Herausgegeben von Robert Barth, Nadja Böller, Urs Dahinden, Sonja Hierl
und Hans-Dieter Zimmermann
Die Lernende Bibliothek 2009
Aktuelle Herausforderungen für die Bibliothek und ihre Partner im Prozess des
wissenschaftlichen Arbeitens
Chur, 2009
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 34
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Hans-Dieter Zimmermann
Rene Frei
Die Informationswissenschaft aus Sicht des Radikalen Konstruktivismus
Chur, 2009
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 35
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Hans-Dieter Zimmermann
Lydia Bauer, Nadja Böller, Sonja Hierl
DIAMOND Didactical Approach for Multiple Competence Development
Chur, 2009
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 36
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Wolfgang Semar
Michaela Spiess
Einsatz von Competitive Intelligence in Schweizer Spitäler
Chur, 2009
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 37
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Wolfgang Semar
Jasmine Milz
Informationskompetenz-Vermittlung an Deutschschweizer Fachhochschulen:
eine quantitative Inhaltsanalyse der Curricula
Chur, 2010
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 38
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Wolfgang Semar
Corinne Keller
RFID in Schweizer Bibliotheken – eine Übersicht
Chur, 2010
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 39
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Wolfgang Semar
Bibliotheksbau in der Schweiz 1985 – 2010
Planung – Nutzung – Ästhetik
Herausgegeben von Robert Barth und Iris Kuppelwieser
Chur, 2010
ISSN1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 40
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Wolfgang Semar
Stephan Becker
Klassifikationsraster zur Relevanzanalyse aktueller Themenanfragen an einer
Mediendokumentationsstelle in der Schweiz
Chur, 2010
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 41
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Wolfgang Semar
Reihe Berufsmarktforschung – Arbeitsbericht 5:
Iris Capatt, Urs Dahinden
Absolventenbefragung 2010
Bachelorstudiengang Informationswissenschaft und Diplomstudiengang Information und Dokumentation der HTW Chur
Chur, 2010
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 42
Herausgegeben von Robert Barth, Nadja Böller, Sonja Hierl und Wolfgang Semar
Saro Adamo Pepe Fischer
Bestandserhaltung im Film-/Videoarchiv des Schweizer Fernsehens
Chur, 2010
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 43
Herausgegeben von Robert Barth, Iris Capatt, Sonja Hierl und Wolfgang Semar
Patricia Dürring
Ökonomischer Mehrwert von Bibliotheken, aufgezeigt anhand ausgewählter Dienste der Zentral- und Hochschulbibliothek Luzern
Chur, 2011
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 44
Herausgegeben von Robert Barth, Iris Capatt, Sonja Hierl und Wolfgang Semar
Pia Baier Benninger
Model Requirements for the Management of Electronic Records (MoReq2).
Anleitung zur Umsetzung
Chur, 2011
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 45
Herausgegeben von Robert Barth, Iris Capatt, Sonja Hierl und Wolfgang Semar
Martina Thomi
Überblick und Bewertung von Musiksuchmaschinen
Chur, 2011
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 46
Herausgegeben von Robert Barth, Iris Capatt und Wolfgang Semar
Regula Trachsler
Angebote für Senioren in Deutschschweizer Bibliotheken
Chur, 2011
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 47
Herausgegeben von Robert Barth, Iris Capatt und Wolfgang Semar
Wolfgang Semar (Hrsg.)
Arge Alp Tagung 23.-24. September 2010, Chur
Informationsgesellschaft und Infrastrukturpolitik im Alpenraum
Chur, 2011
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 48
Herausgegeben von Robert Barth, Lydia Bauer, Iris Capatt und Wolfgang Semar
Heinz Mathys
Jungs lesen weniger als Mädchen.
Was können Bibliotheken gemeinsam mit den Schulen tun, um dies zu ändern?
Chur, 2011
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 49
Herausgegeben von Robert Barth, Lydia Bauer, Iris Capatt und Wolfgang Semar
Anina Baumann
Stärken und Schwächen von Discovery Diensten am Beispiel des EBSCO Discovery Service
Chur, 2011
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 50
Herausgegeben von Robert Barth, Lydia Bauer, Iris Capatt und Wolfgang Semar
Reihe Berufsmarktforschung – Arbeitsbericht 6:
Iris Capatt, Urs Dahinden
Absolventenbefragung 2011
Hochschule für Technik und Wirtschaft HTW Chur Weiterbildungsstudiengänge Informationswissenschaft.
Externer Bericht.
Chur, 2011
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 51
Herausgegeben von Robert Barth, Lydia Bauer, Iris Capatt und Wolfgang Semar
Reihe Berufsmarktforschung – Arbeitsbericht 7:
Iris Capatt, Urs Dahinden
Absolventenbefragung 2011
Hochschule für Technik und Wirtschaft HTW Chur Weiterbildungsstudiengänge Management.
Externer Bericht.
Chur, 2011
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 52
Herausgegeben von Robert Barth, Lydia Bauer, Iris Capatt und Wolfgang Semar
Salome Arnold
Auf den Spuren der Barrieren für ein barrierefreies Webdesign
Chur, 2011
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 53
Herausgegeben von Robert Barth, Lydia Bauer, Iris Capatt und Wolfgang Semar
Laura Stadler
Die Gläserne Decke in Schweizer Bibliotheken
Chur, 2012
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 54
Herausgegeben von Robert Barth, Lydia Bauer, Brigitte Lutz und Wolfgang Semar
Ruth Süess
Evaluation von Web Monitoring Tools zur softwaregestützten Informationsbeschaffung am Beispiel ausgewählter Open Source Web Monitoring Tools
Chur, 2012
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 55
Herausgegeben von Robert Barth, Lydia Bauer, Brigitte Lutz und Wolfgang Semar
Michael Hunziker
Approval Plans und andere Outsourcing-Formen im Bestandesaufbau an den Wissenschaftlichen Bibliotheken der Deutschschweiz
Chur, 2012
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 56
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Urs Dahinden, Michael Aschwanden und Lydia Bauer
Verpasste Chancen? Altersspezifische digitale Ungleichheiten bei der Nutzung von Mobilkommunikation und Internet
Chur, 2012
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 57
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Grégoire Savary
Eine Konservierungsstrategie für das Archiv der Siedlungsgenossenschaft Freidorf bei Muttenz. Eine Hilfestellung für kleine Archive mit gemischten Beständen
Chur, 2013
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 58
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Patrick Wermelinger
Die Georeferenzierung von Katalogdaten mit Hilfe von Linked Open Data
Chur, 2013
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 59
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Carla Biasini
E-Books in öffentlichen Bibliotheken der Schweiz – Determinanten der Akzeptanz bei Kunden
Chur, 2013
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 60
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Nadja Böller
Modell zur strategischen Analyse von Konzepten zur Förderung der Informationskompetenz durch Hochschulbibliotheken – MOSAIK-PRO
Chur, 2013
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 61
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Nina Santner
Von der Mediothek zum Recherchezentrum
Chur, 2013
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 62
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Daniela Denzer
Gründe für die Nichtnutzung von Bibliotheken bei Pensionierten in der Deutschschweiz
Chur, 2013
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 63
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Verena Gerber-Menz
Übernahme von born-digital Fotobeständen und Fotografennachlässen ins Archiv
Chur, 2014
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 64
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Vanessa Kellenberger
E-Shop Analytics und Erfolgsoptimierung – Die wichtigsten Kennzahlen
Chur, 2014
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 65
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Matthias Dudli
Open Innovation in Bibliotheken – Eine Konzeptstudie der ETH-Bibliothek Zürich
Chur, 2014
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 66
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Sarah Carbis
Welche Verbandszeitschrift wünschen sich die Mitglieder des BIS?
Chur, 2014
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 67
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Yvonne Lingg
Patientenverfügung als Informations- und Kommunikationsinstrument
Analyse der Vielfalt sowie Dokumentation der Inhalte und Standardisierungsmöglichkeiten
Chur, 2014
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 68
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Mara Sophie Hellstern
Förderung von Engagement in GLAM (Galleries, Libraries, Archives and Museums) durch Wikipedians in Residence (WiR)
Chur, 2014
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 69
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Philipp Trottmann
Die epochale Trendwende: Der Benutzerrückgang an öffentlichen Bibliotheken der Deutschschweiz
Chur, 2014
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 70
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Ursula Huber
10 Jahre Open Access Initiative – Eine Zwischenbilanz für die Schweiz
Chur, 2014
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 71
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Beat Mattmann
Die Möglichkeiten von RDA bei der Erschliessung historischer Sondermaterialien
Chur, 2014
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 72
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Diane Golay
User-center redesign of the Biotechgate portal: a remote usability testing case study
Chur, 2015
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 73
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Felicitas Isler
Inklusion von Mitarbeitenden mit einer Beeinträchtigung in Bibliotheken
Chur, 2015
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 74
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Tamara Müller
Die Schwierigkeiten bei der Recherche im Archiv(-katalog): Ursachenforschung und Vorschläge zur Problembehebung
Chur, 2015
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 75
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Benjamin Fischer
Potential von automatischen Videoanalysen im Fussball am Beispiel der Schweizer Super League
Chur, 2015
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 76
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Simon Schultze
Videospielturniere in öffentlichen Schweizer Bibliotheken
Ein Pilotprojekt der St. Galler Stadtbibliothek Katharinen
Chur, 2015
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 77
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Charlotte Frauchiger
Barrierefreie E-Books
Chur, 2016
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 78
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Stefanie Dietiker
Cognitive Map einer Bibliothek
Eine Überprüfung der Methodentauglichkeit im Bereich Bibliothekswissenschaft –
am Beispiel der Kantonsbibliothek Graubünden
Chur, 2016
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 79
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Sharon Alt
Konzeption und Evaluation eines Online-Tutorial zur Förderung der
E-Health-Literacy von Männern im Alter von 50 bis 80 Jahren
Chur, 2016
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 80
Herausgegeben von Wolfgang Semar und Brigitte Lutz
Bettina Wille
Automatisierung und Digitalisierung in den wissenschaftlichen Bibliotheken der Schweiz
Ein Oral History Projekt
Chur, 2016
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 81
Herausgegeben von Wolfgang Semar
Michael Mente
Ansichtskarten sind Ansichtssache – Bilder, Grüsse und Metadaten
Über den Wert topografischer Ansichtskarten in Archivbeständen und
Einsichten in Fragen ihrer archivischen Erschliessung
Chur, 2016
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 82
Herausgegeben von Wolfgang Semar
Fabian Muster
Datenstrategiemodell: Ein Referenzmodell zur Entwicklung von Datenstrategien
Chur, 2016
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 83
Herausgegeben von Wolfgang Semar
Sandro Lorenzo
Bibliotheken und Integration
Aspekte der interkulturellen Bibliotheksarbeit und deren Einfluss auf die Integration von
Migranten und Migrantinnen sowie Menschen mit Migrationshintergrund in der Deutschschweiz
mit einem Fokus auf den deutschsprachigen Teil des Kantons Bern
Chur, 2016
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 84
Herausgegeben von Wolfgang Semar
Johannes Reitze
Was öffentliche Bibliotheken meinen, wenn sie vom Dritten Ort sprechen
Chur, 2016
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 85
Herausgegeben von Wolfgang Semar
Simone Beeler
Sonntagsöffnungszeiten in öffentlichen Bibliotheken in der Schweiz
Chur, 2017
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 86
Herausgegeben von Wolfgang Semar
Marco Humbel
Die Umsetzung von Open Data an Wissenschaftlichen Bibliotheken der Schweiz:
Eine qualitative Untersuchung
Chur, 2017
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 87
Herausgegeben von Wolfgang Semar
Flurina Huonder
Medieninhaltsanalyse Big Data:
Big Data, Datenschutz und Privatsphäre in Schweizer und US-amerikanischen Zeitungen
Chur, 2017
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 88
Herausgegeben von Wolfgang Semar
Marcel Hanselmann
Makerspaces in öffentlichen Bibliotheken:
Eine Untersuchung der didaktischen Ziele und eine Evaluation der Technologie littleBits
Chur, 2017
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 89
Herausgegeben von Wolfgang Semar
Franziska Brunner
Überlieferungsbildung 2.0:
Eine Untersuchung zum Mehrwert von Partizipation Dritter in staatlichen Archiven
Chur, 2017
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 90
Herausgegeben von Wolfgang Semar
Marcella Haab-Grothof
„Kleider machen BibliothekarInnen“:
Der Einfluss von Kleidung des Bibliothekspersonals auf die Kontaktaufnahme von Benutzenden
Chur, 2017
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 91
Herausgegeben von Wolfgang Semar
Sven Lenz
Customer Engagement Analytics: Clustering User Navigation Behaviour
Chur, 2017
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 92
Herausgegeben von Wolfgang Semar
Isabel Merlo
Projektmanagement in Schweizer Bibliotheken
Eine Untersuchung, wie Schweizer Bibliotheken Projekte managen und ein
Projektmanagementvorschlag für die GGG Stadtbibliothek Basel
Chur, 2018
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 93
Herausgegeben von Wolfgang Semar
Silvana Rüfli
Die Usability von E-Book-Angeboten wissenschaftlicher Bibliotheken
Eine Untersuchung am Beispiel der Universitätsbibliotheken
St. Gallen, Bern und Zürich
Chur, 2018
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 94
Herausgegeben von Wolfgang Semar
Vera Knoll
Leichte Sprache in amtlichen Publikationen und Webseiten
Wie ernst nehmen Verwaltungen die Leichte Sprache in der deutschsprachigen Schweiz?
Chur, 2018
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 95
Herausgegeben von Wolfgang Semar
Andrea Traber
Wie lernen studentische Bibliotheks-Nutzende und was macht für sie den optimalen Arbeitsplatz aus?
Eine Studie der Lernlandschaft der Universitätsbibliothek St. Gallen
Chur, 2018
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 96
Herausgegeben von Wolfgang Semar
Irina Morell
„Für das Volk und durch das Volk?“
Bibliotheken als Gegenstand von Volksabstimmungen und Petitionen
Chur, 2018
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 97
Herausgegeben von Wolfgang Semar
Monika Rohner
Betrachtung der Data Visualization Literacy in der angestrebten Schweizer Informationsgesellschaft
Chur, 2018
ISSN 1660-945X

Churer Schriften zur Informationswissenschaft – Schrift 98
Herausgegeben von Wolfgang Semar
Kirsten Scherer Auberson
Counteracting Concept Drift in Natural Language Classifiers:
Proposal for an Automated Method
Chur, 2018
ISSN 1660-945X

Über die Informationswissenschaft der HTW Chur

Die Informationswissenschaft ist in der Schweiz noch ein relativ junger Lehr- und Forschungsbereich. International weist diese Disziplin aber vor allem im anglo-amerikanischen Bereich eine jahrzehntelange Tradition auf. Die klassischen Bezeichnungen dort sind Information Science, Library Science oder Information Studies. Die Grundfragestellung der Informationswissenschaft liegt in der Betrachtung der Rolle und des Umgangs mit Information in allen ihren Ausprägungen und Medien sowohl in Wirtschaft und Gesellschaft. Die Informationswissenschaft wird in Chur integriert betrachtet.

Diese Sicht umfasst nicht nur die Teildisziplinen Bibliothekswissenschaft, Archivwissenschaft und Dokumentationswissenschaft. Auch neue Entwicklungen im Bereich Medienwirtschaft, Informations- und Wissensmanagement und Big Data werden gezielt aufgegriffen und im Lehr- und Forschungsprogramm berücksichtigt.

Der Studiengang Informationswissenschaft wird seit 1998 als Vollzeitstudiengang in Chur angeboten und seit 2002 als Teilzeit-Studiengang in Zürich. Seit 2010 rundet der Master of Science in Business Administration das Lehrangebot ab.

Der Arbeitsbereich Informationswissenschaft vereinigt Cluster von Forschungs-, Entwicklungs- und Dienstleistungspotenzialen in unterschiedlichen Kompetenzzentren:

- Information Management & Competitive Intelligence
- Collaborative Knowledge Management
- Information and Data Management
- Records Management
- Library Consulting
- Information Laboratory

Diese Kompetenzzentren werden im **Swiss Institute for Information Research** zusammengefasst.

IMPRESSUM

Verlag & Anschrift

Arbeitsbereich Informationswissenschaft

HTW - Hochschule für Technik und Wirtschaft
University of Applied Sciences
Ringstrasse 37
CH-7000 Chur

www.informationswissenschaft.ch

www.htwchur.ch

ISSN 1660-945X

Institutsleitung

Prof. Dr. Niklaus Stettler
Telefon: +41 81 286 24 61
Email: niklaus.stettler@htwchur.ch

Sekretariat

Telefon : +41 81 286 24 24
Fax : +41 81 286 24 00
Email: clarita.decurtins@htwchur.ch