



UNIVERSITÄT ZU LÜBECK

From the Institute of Information Systems
of the University of Lübeck
Director: Prof. Dr.-Ing. Nele Rußwinkel

Toward better Subjective Content Descriptions: A Spiral of Estimation, Enrichment, Usage, and Improvement

Dissertation
for Fulfilment of
Requirements
for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences and Technical Engineering

Submitted by

Magnus Bender
from Lübeck

Lübeck 2024

First referee: Prof. Dr. rer. nat. habil. Ralf Möller
Second referee: Prof. Dr. rer. nat. Diedrich Wolter
Date of oral examination: October 2, 2024
Approved for printing. Lübeck, October 18, 2024

Abstract

In today’s modern world, humans not only interact with each other, but also with various intelligent systems, which are composed of artificial agents acting on behalf of human principals. Natural language is the most intuitive form of human communication, and the assumption is that humans would like to apply natural language when interacting with agents. However, despite recent advances in human-machine interaction, natural language processing is known to be quite complex for machines and difficult to realize effectively.

In this dissertation, the goal is to enable agents to communicate with humans about natural language texts while considering the context a human principal is working in. To achieve this goal, today’s services such as standard Information Retrieval (IR) or Large Language Models (LLMs) are not sufficient. In this dissertation, we consider “comments” attached with text documents in the corpus. These “comments” might have no direct grounding in the base text, but indeed involve the context of principals. If “comments” are formulated in natural language, a human principal can easily be pointed to relevant documents and, in those documents, to paragraphs or sentences that are relevant for the context, e.g., a new scientific problem the human is working on.

The “comments” attached with text documents are considered to be *subjective* because it is the context, not only the base text, that matters for the interpretation of comments. Thus, we call the above-mentioned “comments” Subjective Content Descriptions (SCDs). A principal can specify the context in two ways, first using SCDs associated with the corpus, and second as part of queries to the agent. Another advantage of using SCDs for text understanding is that techniques using SCDs require less computational resources than LLMs and run efficiently on off-the-shelf hardware.

This dissertation contributes techniques required by an SCD-based IR agent. Such an agent is characterized by the ability to consider the context and *subjectivity* of its principals. First, for a corpus, an initial set of SCDs including labels needs to be estimated by the agent. Afterwards, the SCDs need to be improved, e.g., by human feedback. Updating SCDs is also necessary if the context or principal changes. Finally, we consider relations between SCDs, e.g., complementarity. We define how to measure complementarity of SCDs and enable the IR agent to use complementarity.

Kurzfassung

In der heutigen Welt interagieren Menschen nicht nur miteinander, sondern auch mit intelligenten Systemen, die aus Agenten bestehen und im Auftrag von Menschen handeln. Natürliche Sprache ist die intuitivste Form menschlicher Kommunikation, und es ist anzunehmen, dass Menschen bei der Interaktion mit Agenten natürliche Sprache bevorzugen. Trotz jüngster Fortschritte im Bereich der Mensch-Maschine-Interaktion ist die Verarbeitung natürlicher Sprache für Maschinen jedoch bekanntermaßen komplex und schwer zu realisieren.

In dieser Dissertation geht es darum, Agenten zu ermöglichen, mit Menschen über natürlichsprachliche Texte zu kommunizieren und dabei den Kontext zu berücksichtigen, in dem sich ein Mensch gerade befindet. Heutige Dienste zum standardmäßigen Information Retrieval (IR) oder große Sprachmodelle (LLMs) sind jedoch nicht ausreichend, um dieses Ziel zu erreichen. Wir betrachten in dieser Dissertation „Kommentare“ verbunden mit Textdokumenten im Korpus. Diese „Kommentare“ haben möglicherweise keinen direkten Bezug zum Text und erschließen sich erst durch den Kontext. Wenn „Kommentare“ in natürlicher Sprache formuliert sind, kann ein Mensch dadurch leicht auf relevante Dokumente und darin auf Absätze oder Sätze verwiesen werden, jeweils bezogen auf den Kontext, wie z. B. ein ganz neues wissenschaftliches Problem.

Diese einem Textdokument beigelegten „Kommentare“ sind *subjektiv*, da der Kontext und nicht nur das Textdokument selbst für die Interpretation entscheidend ist. Daher nennen wir diese „Kommentare“ subjektive Inhaltsbeschreibungen (SCDs). Ein Mensch kann den Kontext nun zweifach spezifizieren, zum einen mit Hilfe von SCDs, die mit dem Korpus verbunden sind, und zum anderen als Teil der Anfragen an den Agenten. Ein weiterer Vorteil von SCDs ist, dass Techniken für SCDs weniger Ressourcen als LLMs benötigen und effizient auf Standard-Hardware laufen.

Diese Dissertation stellt Techniken vor, die für einen SCD-basierten IR-Agenten benötigt werden. Ein solcher Agent zeichnet sich durch die Fähigkeit aus, den Kontext und die Subjektivität von Menschen zu berücksichtigen. Zunächst muss der Agent für einen Korpus initiale SCDs einschließlich Labels bestimmen. Anschließend werden die SCDs, z. B. durch menschliches Feedback, verbessert. Eine Aktualisierung der SCDs ist auch erforderlich, wenn sich Kontext oder Menschen verändern. Schließlich gibt es noch Relationen zwischen SCDs, z. B. die Komplementarität. Wir zeigen, wie solche Relationen bestimmt und im IR-Agenten genutzt werden können.

Acknowledgements

Working on a dissertation is a demanding and sometimes exhausting task. In the end, however, you can see that a result has emerged and sometimes this comes faster as initially thought.

So, thank you, Ralf Möller, for giving me this opportunity! Thank you for the numerous discussions about my and other's research, but also about artificial intelligence in general. Often we had to discuss matters concerning teaching, too. You always had an open ear for my ideas, questions, and issues—guiding me with advices through the journey of my dissertation. I also would like to thank Diedrich Wolter for reviewing my dissertation and Thomas Eisenbarth for chairing my defense.

It all started during my bachelor's studies at University of Lübeck. Tanya Braun asked me if I wanted to become a tutor of "Introduction to Web and Data Science". I said "Yes" and my interest in the topic of data science started to grow. Thank you, Tanya. Looking for a topic of my bachelor's thesis, I asked Tanya and Ralf, and chose a topic about Latent Dirichlet Allocation. It was a great time and I learned a lot writing this bachelor's thesis—my first academic work. Thank you, Felix Kuhr, for introducing me to the academics. After finishing my bachelor, I was able to support Felix with his evaluations for his dissertation. Hence, I kept working with Felix and Tanya on the field of text understanding during my master's studies. Subjective Content Descriptions were also born during this time.

When the master's thesis was due, I naturally went to ask Ralf for a topic. Thus, I completed my master doing research with Felix, Tanya, and Ralf. Directly afterwards I started as doctoral student at the Institute of Information Systems with Ralf becoming my supervisor. I was lucky enough to work with Subjective Content Descriptions again.

The work at the institute mostly consists of two parts: Doing research for the dissertation and teaching—at least in a simplified world. Thank you, Malte Luttermann, for going through bachelor and master with me. You became a doctoral student of Ralf, too. Our very good and efficient collaboration could go on, even if it didn't fit with the topic of the dissertation, it did with teaching. Thank you, Marcel Gehrke, for taking me by hand writing my first *own* paper, giving me feedback to multiple versions of each following paper, and discussing new ideas. Several times I came out

of a conversation about my research with the clear idea and solution in mind—only being required to \LaTeX it. However, it is slightly more than that.

My thanks also go to the team members of the institute for all the other discussions and conversations. I would also like to thank Angela König and Nils Fußgänger for helping me with all the organizational topics and keeping many of them away from me.

Working on a dissertation is somehow a cycle of having an idea or problem, formalizing it, working on a solution, doing an evaluation or proof, and summarizing everything in a paper. When a paper is finished, the next one is ready to come. So, this is similar to a spiral going up—each paper builds on the previous and runs through the same steps. It reminds me of a serpentine road going up a mountain. Finishing this dissertation, a plateau is reached, but there are more plateaus further up and other mountains, too.

Thank you to everyone being on this journey with me!

Magnus Bender
Lübeck, February & October 2024

Contents

Abstract	iii
Kurzfassung	v
Acknowledgements	vi
List of Variables, Notations, and Abbreviations	xii
1. Introduction	1
1.1. Related Work	2
1.2. Overview of Contributions	6
1.3. Structure	7
2. Preliminaries	9
2.1. Notations for Corpora	9
2.2. Natural Language Processing Techniques	10
2.3. Intelligent Agents	13
3. The Universe of Subjective Content Descriptions	15
3.1. Subjective Content Descriptions	16
3.2. SCDs in an Information Retrieval Agent	19
3.3. Detailed Contributions	23
I. Theoretical Foundation	27
4. USEM – UnSupervised Estimation of SCDs	29
4.1. Introduction	29
4.2. Unsupervised Estimation of SCDs	31
4.3. Evaluation	37
4.4. Related Work	42
4.5. Interim Conclusion	44

5. LESS is More – Label Estimation for SCDs without Supervision	45
5.1. Introduction	45
5.2. Computing Labels for SCDs	47
5.3. Evaluation	50
5.4. Interim Conclusion	55
6. FrESH – Feedback-reliant Enhancement of SCDs by Humans	57
6.1. Introduction	57
6.2. Incorporate Feedback	58
6.3. Evaluation	61
6.4. Interim Conclusion	64
7. ReFrESH – Relation-preserving Feedback-reliant Enhancement of SCDs	65
7.1. Introduction	65
7.2. Related Work	66
7.3. Relation-preserving Updates on SCD Matrices	67
7.4. Evaluation	73
7.5. Interim Conclusion	77
8. Complementarity as an Inter-SCD Relation	79
8.1. Introduction	79
8.2. Related Work	81
8.3. Preliminaries	82
8.4. Identifying Complementary Documents	83
8.5. Document Classification with Complementarity and Similarity	90
8.6. Evaluation	98
8.7. Interim Conclusion	104
II. Application	107
9. Composing an Information System using SCDs	109
9.1. Introduction	109
9.2. Basic Structure	110
9.3. Working with Corpora	112
9.4. Working with SCDs	114
10. SCDs in Further Domains	123
10.1. Introduction	123
10.2. Humanities Aligned Chatbot	124
10.3. Research Data Repository Integration	127

11. Conclusion	131
11.1. Summary of Contributions	131
11.2. Outlook	133
A. Appendix	135
Bibliography	137
Publications	146
Curriculum Vitae	150

List of Variables, Notations, and Abbreviations

This list provides an overview of the basic and reoccurring variables, notations, and abbreviations used in this dissertation. All items are listed in order of appearance and are introduced in the chapters of the dissertation, too.

• Chapter 1 – General Abbreviations

NLP : Natural Language Processing

AI : Artificial Intelligence

IR : Information Retrieval

SCD : Subjective Content Description

LLM : Large Language Model

LDA : Latent Dirichlet Allocation

BERT : Bidirectional Encoder Representations from Transformers

GPT : Generative Pre-Trained Transformer

IE : Information Extraction

ChatHA : Humanities Aligned Chatbot

RDR : Research Data Repository

• Chapter 2 – Notations for Corpora

w : Word from given vocabulary $\mathcal{V} = \{w_1, \dots, w_L\}$, $L \in \mathbb{N}$

s : Sentence, sequence of words $s = (w_1, \dots, w_N)$, $N \in \mathbb{N}$, $w_i \in \mathcal{V}$

d : Document, a sequence of sentences $d = (s_1^d, \dots, s_{M^d}^d)$, $M^d \in \mathbb{N}$

\mathcal{D} : Corpus of documents $\mathcal{D} = \{d_1, \dots, d_D\}$, $D \in \mathbb{N}$

M : Number of sentences in corpus, $M = \sum_{d \in \mathcal{D}} M^d$

\mathcal{C} : SCD's additional content, e.g., label l and relations R

-
- t : SCD, tuple $t = (\mathcal{C}, \{s_1, \dots, s_S\})$, $S \in \mathbb{N}$
 $g(\mathcal{D})$: SCD set for corpus \mathcal{D}
 $g(d)$: SCD set for document d
 $I(w_i, s^d)$: Influence value for each word w_i in sentence s^d
- **Chapter 3** – Variables and notations used with SCDs

K : Number of SCDs for corpus, $K \in \mathbb{N}$
 L : Size of vocabulary \mathcal{V} , $L \in \mathbb{N}$
 $\delta(\mathcal{D})$: SCD matrix for \mathcal{D} , shaped $K \times L$
 $(v_{i,1}, \dots, v_{i,L})$: i -th row of SCD matrix $\delta(\mathcal{D})$, i.e, word distribution of t_i
 $\delta(s)$: Word count vector of sentence s , shaped $1 \times L$
 \vec{s} : Word count vector of sentence s , shaped $1 \times L$
 \mathcal{W} : Sequence of MPS²CD similarity values $sim \in \mathcal{W}$
 - **Chapter 3** – Vocabulary and techniques used with SCDs

SCD : Subjective Content Description
 SCD matrix : SCD-word distribution matrix
 SEM : Supervised Estimator of SCD Matrices
 MPS²CD : Most Probably Suited SCD
 iSCD : Inline SCD
 cSCD matrix : Combined SCD matrix (related and complementary SCDs)
 - **Part I** – Contributed techniques of this dissertation

USEM : UnSupervised Estimator of SCD Matrices
 LESS : Label Estimation for SCDs without Supervision
 FrESH : Feedback-reliant Enhancement of SCDs by Humans
 ReFrESH : Relation-preserving Feedback-reliant Enhancement of SCDs
 - **Part II** – Applications of the techniques of this dissertation

SIS : SCD-based Information System
 ChatHA : Humanities Aligned Chatbot

1. Introduction

In today's modern world, humans not only interact with each other, but also with various intelligent systems, which are composed of artificial agents acting on behalf of human principals. Natural language is the most intuitive form of human communication, and the assumption is that humans would like to apply natural language when interacting with agents. For example, humans give task descriptions to artificial agents, which then carry out the task without the requirement to be micromanaged. We assume that agents report their task results back to their human principals. However, despite recent advances in human-machine interaction, Natural Language Processing (NLP) is known to be quite complex for machines and difficult to realize effectively. The complexity persists even when focussing on just one language, such as, e.g., English. Based on a task description, agents are expected to make decisions and execute actions to fulfil their respective task descriptions. In addition, agents continuously reinterpret their task descriptions based on the developing interaction context specified in natural language.

NLP is an important field of research that bridges computer science, linguistics, and Artificial Intelligence (AI). In the context of this dissertation, the goal of NLP is to enable agents to communicate with humans about natural language texts while dealing with task descriptions in a way that appropriately considers the context the human is working in. To achieve this goal, today's services such as standard Information Retrieval (IR), summarization services, or writing style enhancements, to name just a few common services, are not sufficient.

Applying the idea of this dissertation, for instance, on IR returning a set of text documents or summarization returning (rather large) texts, agents should be able to comment on those returned texts from the perspective of the context in which a human principal works. This involves the attachment of "comments" which might have no direct grounding in the base text, but indeed require the context to make sense. If "comments" are formulated in natural language, a human principal can easily be pointed to relevant documents and, in those documents, to paragraphs or sentences that are relevant for the context, e.g., a new scientific problem the human is working on. The conjecture of this dissertation is that "comments" ensure that the human principal can easily grasp why a document, paragraph, or sentence is relevant for the context the human works on without requiring the human to read all parts of the texts.

To illustrate these rather abstract ideas, think about studying for an exam. There are scripts, slides, and additional materials such as books that cover the topic of the exam. These text documents form a corpus, which is used by the agent to perform IR. The agent’s principal is in a specific context, i.e., studying for the exam. An important question that this dissertation investigates is how can the context be made accessible to the agent as part of the task description?

Specifically, the context while studying for an exam comprises, e.g., the level of knowledge across the different areas in the topic of the exam or the personal interest for the topic. This context is available to the agent through the “comments” attached to the documents in the corpus. Some “comments” have been added by the principal during lectures, e.g., notes, examples, or references to other materials. Also questions and hints for understanding are added as “comments”. Based on the number of “comments” each text document has, the agent is able to estimate the principal’s interest in this document. In addition, text documents with many “comments” containing questions may indicate an area in which the principal is less proficient. In summary, when studying for an exam, context-specific “comments” help the agent to be beneficial to its principal, e.g., by identifying less proficient areas and highlighting references to more proficient areas to support understanding.

The “comments” attached with text documents are considered to be *subjective* because it is the context, not the base text, that matters for the interpretation of comments. Thus, we call the above-mentioned “comments” in this dissertation Subjective Content Descriptions, or SCDs for short. The reader might imagine SCDs as sticky notes attached to a certain part of the text documents and filled with paratext to reflect the *subjectivity* of the human principal. SCDs can also contain references to data or other texts. We first illuminate these ideas in the light of a discussion about related work in NLP, and then summarize the main contributions of this dissertation.

1.1. Related Work

This section starts with a short characterization of NLP including the currently very popular Large Language Models (LLMs) and continues with text annotation. Thereby, we focus on the subjectivity brought by humans and the context created by interaction with an NLP-based agent. Finally, we outline open problems in the field of SCDs.

In addition to this section of related work, multiple chapters of this dissertation have their own section of related work especially relevant for the actual chapter.

1.1.1. Symbolic and Statistical Approaches

NLP systems can be roughly divided into systems using symbolic and systems using statistical approaches. Recently, neural networks are popular for use in NLP systems.

A symbolic approach is characterized by the idea to identify symbols like letters or words and transform them based on rules. Hence, symbolic systems often consist of a large amount of universal rules which are applied to natural language text. The rules are often hand crafted, which is very time-consuming and does not consider *subjectivity*.

Statistical approaches have in common that they use statistics of corpora of natural language text, e.g., count words in sentences, documents, and the entire corpus. Thereby, statistical NLP tackles the problem of manually creating rules for symbolic NLP. The term-frequency inverse-document-frequency (tf.idf) [SJ72] is a well known and simple approach to calculate the relevance of words in a text document related to its corpus. However, tf.idf calculates relevances only based on words and documents, and does not take context into account in the way we need.

Another technique from the field of statistical NLP are topic models. Topic models represent the topics of a corpus and the topics of the documents in the corpus by distributions over the words. One well-known algorithm to create topic models is Latent Dirichlet Allocation (LDA) [BNJ03]. LDA has many extensions, e.g., the author-topic model [RZGSS04], which enables LDA to observe each author of a document, and the dynamic topic model [BL06], which allows for analyzing topic changes over time. Overall, by LDA topic models are objectively calculated for a certain corpus.

Neural networks are a commonly used technique in the area of machine learning. The structure of a neural network consists of multiple layers, where each layer consists of multiple units, often perceptrons [Ros58]. Together, all the layers implement a function which transforms raw input values to an output value. Then, the output value can be easily processed, e.g., used for making a decision based on a threshold. Neural networks are trained for a specific task pre-defined by a large amount of training data. A well known neural network based NLP technique is Word2Vec [MCCD13]. Word2Vec is a so-called embedding technique, i.e., it takes words as input and encodes these words as vectors in a vector space. Similarly, tf.idf is an embedding technique where the vector space is hand-crafted, unlike Word2Vec where it is trained. In both cases, the output vectors can be, e.g., used to identify similarities or to predict words.

Next, we consider LLMs, which are huge networks used for NLP.

1.1.2. Large Language Models

LLMs are mostly based on the transformer architecture introduced by Vaswani et al. [VSP⁺17]. Thereby, the big advantage of the transformer architecture is that all inputs are simultaneously fed into the model. Thus, the models can be trained fast and in parallel.

Previously, commonly used techniques in the area of NLP were Recurrent Neural Networks (RNNs) [RHW86] and neural networks with Long Short-Term Memory (LSTM) [HS97] units. For both techniques, the input values are fed into the model sequentially one after the other. Inside the network, the previous inputs are then stored by the feedback loops of the RNN or in the LSTM units. When adding a new input the influence of each previous input becomes a bit smaller and vanishes for long input sequences, which is called *vanishing gradient*. Thus, the length of the input is limited in terms of vanishing gradients and model's overall size in terms of training time.

These limits regarding input length and model size are lifted by the transformer architecture. Hence, the introduction of the transformer architecture initiated the development of a large number of new and very large language models.

A well-known LLM is the Bidirectional Encoder Representations from Transformers (BERT) [DCLT19] introduced by Devlin et al. BERT is an encoder and encodes a sequence of inputs, i.e., words, into a sequence of vector representations. Another LLM is the Generative Pre-Trained Transformer (GPT) [RN18, RWC⁺19, BMR⁺20], mainly developed by OpenAI and released in several improved versions. GPT is a generative model, i.e., the model generates natural language. GPT generates a sentence by extending it word by word, and thus, is able to write text based on previous or initial words. State-of-the-art chatbots like ChatGPT¹ or Gemini² use the transformer architecture.

These chatbots process natural language queries as input and respond with natural language output. For many NLP related tasks, it is possible to simply describe the task using natural language and provide the data to the chatbot, which then directly solves the task. Hence, LLMs are becoming more and more a *Swiss knife* for NLP. Challenges arising with complex tasks and requiring several steps are addressed by Auto- and AgentGPT [YYH23].

LLM-based chatbots allow their users to use natural language and also consider the user's context, as long the user adds the context to the query. It is also possible to provide larger amounts of data to the chatbot and thus concretize the context further. However, the LLM used by the chatbot is still trained without a specific

¹<https://chat.openai.com/>

²<https://gemini.google.com/>, formerly Bard

context and mostly responds by using known contexts that are similar to the user’s context. Along with SCDs, LLMs may be enriched with *subjectivity*. Summarized, pure LLMs as well as symbolic and statistical approaches do not fulfil our goal of considering *subjectivity* and contexts brought by humans and users.

1.1.3. Text and Corpus Annotation

Annotations are data associated with locations in text documents of a corpus. Annotating corpora of text documents with additional information has been investigated for a long time. The corresponding subfield of NLP is text and corpus annotation.

The Brown Corpus [Mav69] is one of the first corpora used to analyze natural language. Initially, the distribution of words among different categories and contexts of natural language was analyzed. Later, *part-of-speech* tags were added, these tags can already be interpreted as annotations assigning a class to each word.

In the beginning of natural language annotation, most annotations had to be manually added to the corpora. Today, crowdsourcing can be used to manually annotate text documents [SBDS14]. However, manually adding annotations is a time consuming task. Thus, semi-automatic and automatic annotation systems were developed. Automatic annotation system often use a database of entities and facts, e.g., DBpedia [ABK⁺07] or YAGO [SKW07]. Then, the text documents are annotated with links to the known entities in the database.

With the help of annotations it is possible to add “comments” to texts. Thus, it is also possible to represent *subjectivity* and allow humans to add context in form of annotations. However, the previously described approaches for automatically annotating corpora yield objective annotations or users need to manually create their annotations. We address these issues with SCDs.

1.1.4. Subjective Content Descriptions

The main ideas behind SCDs have been developed by Kuhr et al. and are well described in Kuhr’s dissertation [Kuhr22] including several publications. SCDs provide a formalism to add “comments” to texts and consider these “comments” as *subjective* and context specific. Coming back to our example of studying for an exam, a principal is able to specify the context to the agent using SCDs associated with text documents in the corpus.

Kuhr assumes that a set of initial SCDs exists for a corpus or creates SCDs transitionally via Information Extraction (IE). The SCDs of a corpus specify a first context to consider during IR using SCDs. Each query sent to the IR system gives

a second more detailed context, which is considered, too. However, an initial set of SCDs often does not exist for a corpus, or if there is a set, it is necessary to keep in mind that the SCDs are *subjective*. So different sets of SCDs would be needed for different principals or as the context progresses.

Next, we outline the contributions of this dissertation.

1.2. Overview of Contributions

In this dissertation, we investigate techniques for creating an SCD-based IR agent. Such an agent is characterized by the ability to consider the context and *subjectivity* of its principals, i.e., human users. In the previous section, we identify problems of current approaches and state why SCDs provide useful techniques and formalisms to our problem. However, there are still open problems to solve with this dissertation. Overall, the contributions of this dissertation can be combined and integrated in an SCD-based IR agent accessible through an information system. It is also possible to combine SCDs with an LLM-based chatbot, combining the advances of both techniques.

The first contribution solves the problem of getting an initial set of SCDs for a corpus. After estimating these initial SCDs, the SCDs are mostly sets of similar sentences and thus difficult to grasp or describe to humans. Hence, we enrich the SCDs with labels, i.e., textual descriptions. Both steps work in an unsupervised way and are applicable to any user supplied corpus.

- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **Unsupervised Estimation of Subjective Content Descriptions in an Information System** in *International Journal of Semantic Computing*, 2024
<https://dx.doi.org/10.1142/S1793351X24410034>
- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **Unsupervised Estimation of Subjective Content Descriptions** in *17th IEEE International Conference on Semantic Computing (ICSC 2023)*
<https://dx.doi.org/10.1109/ICSC56153.2023.00052>
- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **LESS is More: LEan Computing for Selective Summaries** in *KI 2023: Advances in Artificial Intelligence. Lecture Notes in Computer Science*, Springer
https://dx.doi.org/10.1007/978-3-031-42608-7_1

The second contribution addresses the problem of updating SCDs when the context of a user changes or other users want to use already existent SCDs themselves. Generally, a corpus may contain SCDs which do not represent the context of a

user or do not fulfill the required use-case. Thus, we contribute two techniques to optimize SCDs based on user-supplied feedback.

- Magnus Bender, Kira Schwandt, Ralf Möller, Marcel Gehrke: **FrESH – Feedback-reliant Enhancement of Subjective Content Descriptions by Humans** in *Proceedings of the Humanities-Centred AI (CHAI) Workshop at KI2023, 46th German Conference on Artificial Intelligence, 2023*
<https://ceur-ws.org/Vol-3580/paper3.pdf> (Slides: <https://dx.doi.org/10.25592/uhhfdm.13423>)
- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **ReFrESH – Relation-preserving Feedback-reliant Enhancement of Subjective Content Descriptions** in *18th IEEE International Conference on Semantic Computing (ICSC 2024)* – Best Paper Award
<https://dx.doi.org/10.1109/ICSC59802.2024.00010>

One type of SCD references multiple locations, e.g., sentences, in the corpus whereas all sentences share the same SCD. However, understanding text documents requires more than groups of similar sentences. Hence, we introduce relations between SCDs and use complementarity as an example relation. We define how to measure complementarity of SCDs and enable the IR agent to retrieve complementary documents for its users.

- Magnus Bender, Felix Kuhr, Tanya Braun: **To Extend or not to Extend? Enriching a Corpus with Complementary and Related Documents** in *International Journal of Semantic Computing, 2022*
<https://dx.doi.org/10.1142/S1793351X2240013X>
- Magnus Bender, Felix Kuhr, Tanya Braun: **To Extend or not to Extend? Complementary Documents** in *16th IEEE International Conference on Semantic Computing (ICSC 2022)*
<https://dx.doi.org/10.1109/ICSC52841.2022.00011>

Together, the contributions of this dissertation provide a cycle of estimating, enriching, using, and improving SCDs toward text understanding. Metaphorically, the SCDs of a corpus approach the top of a spiral—iteration after iteration on the cycle, the SCDs improve incrementally.

1.3. Structure

This dissertation consists of eleven chapters structured in two parts. The first three chapters provide the introduction to the domain of this dissertation.

The introduction (Chapter 1) includes general related work, an overview of the contributions, and this section about the dissertation’s structure. Afterwards, Chapter 2 contains general preliminaries especially from the field of NLP. Chapter 3 provides an introduction to SCDs: It describes the use-cases and objectives of SCDs, introduces techniques for using SCDs, and outlines previously solved as well as open problems in the field of SCDs. Chapter 3 is therefore called “The Universe of Subjective Content Descriptions”, it provides a big picture about SCDs and presents and contextualizes the contributions of this dissertation. The end of Chapter 3 leads over to Part I containing the different contributions of this dissertation.

Part I provides the theoretical foundation and is divided into five chapters. It provides the main content of this dissertation:

- Chapter 4: Estimate SCDs in an unsupervised manner for any corpus.
- Chapter 5: Compute labels for SCDs in an unsupervised manner.
- Chapter 6: Incrementally change SCD-based models and delete faulty sentences from corpora.
- Chapter 7: Incrementally update SCD-based models and optimize faulty associations of SCDs and sentences.
- Chapter 8: Add relations among SCDs and consider complementary between SCDs as an example relation.

Part II provides the application of the theory from Part I and contains two chapters. Chapter 9 presents an information system built on SCDs and implements an IR agent accessible through a web application. It allows users to upload their own corpora and provides IR services based on SCDs after processing the corpora. Chapter 10 demonstrates that SCDs can be used in more applications: The Humanities Aligned Chatbot (ChatHA) uses SCDs for processing its output. Using SCDs reduces hallucinations and adds citations to the responses. Furthermore, SCDs can be integrated with a Research Data Repository (RDR) to provide data visualization on demand.

The dissertation concludes with Chapter 11 which provides a summary of the results and an overall conclusion. Furthermore, we provide an outlook to still open problems in the field of SCDs and possible further applications of SCDs.

2. Preliminaries

This chapter introduces general preliminaries for the domain of text understanding as required in this dissertation. First, we formalize corpora of text documents and the parts they are made of. Second, we give a general overview of NLP with a focus on IR, corpus annotations, and topic models—especially LDA. Finally, we introduce agents in the context of AI.

The subsequent sections give the introduction to SCDs and the techniques used while working with SCDs. Additionally, Chapter 8 has its own preliminaries section containing additional preliminaries relevant for this chapter.

2.1. Notations for Corpora

First, we formalize our setting of a corpus of text documents.

- A word w_i is a basic unit of discrete data from a finite vocabulary $\mathcal{V} = \{w_1, \dots, w_L\}$, $L \in \mathbb{N}$.
- A sentence s is defined as a sequence of words $s = (w_1, \dots, w_N)$, $N \in \mathbb{N}$, where each word $w_i \in s$ is an element of vocabulary \mathcal{V} . Commonly, a sentence is terminated by punctuation symbols like “.”, “!”, or “?”.
- A document d is defined as a sequence of sentences $d = (s_1^d, \dots, s_{M^d}^d)$, $M^d \in \mathbb{N}$.
- A corpus \mathcal{D} represents a set of documents $\{d_1, \dots, d_D\}$, $D \in \mathbb{N}$. Thus, the overall number of sentences in a corpus is $M = \sum_{d \in \mathcal{D}} M^d$.
- An SCD t is a tuple of the SCD’s additional data \mathcal{C} and references to the referenced sentences $\{s_1, \dots, s_S\}$, $S \in \mathbb{N}$. Thus, each SCD references sentences in documents of \mathcal{D} , while in the opposite direction a sentence is associated with an SCD. The additional data is a set \mathcal{C} containing, e.g., a label l of the SCD and a set R of relations. Thereby, these relations in R can be considered as links between two SCDs or between an SCD and a sentence. They may also be accompanied with a weight or factor and a type for each link.

- A sentence associated with an SCD is called SCD window, inspired by a tumbling window moving over the words of a document. Generally, an SCD window may not be equal to a sentence, may be a subsequence of a sentence, or the concatenated subsequences of two sentences, too.
- For a corpus \mathcal{D} there exists a set g called SCD set containing K associated SCDs

$$g(\mathcal{D}) = \left\{ \underbrace{\left(\overbrace{\{l_j, R_j, \dots\}}^{c_j}, \bigcup_{d \in \mathcal{D}} \{s_1^d, \dots, s_{S^d}^d\} \right)}_{t_j} \right\}_{j=1}^K.$$

Given a document $d \in \mathcal{D}$, the term $g(d)$ refers to the set of SCDs associated with sentences from document d .

- Each word $w_i \in s^d$ is associated with an influence value $I(w_i, s^d)$ representing the relevance of w_i in the sentence s^d . For example, the closer w_i is positioned to the object of the sentence s^d , the higher its corresponding influence value $I(w_i, s^d)$. The influence value is chosen according to the task and might be modeled with a binomial, linear, or uniform distribution.

2.2. Natural Language Processing Techniques

The field of NLP comprises many techniques. In this section we describe some of them that are used in multiple chapters of this dissertation.

2.2.1. Information Retrieval

In the domain of NLP, IR is about retrieving information from natural language texts. Simplified, IR can be imagined as performing a search in a corpus. A query is sent to an IR system, which then generates an answer and returns it.

The query specifies the information need of a user. Natural language words or text are possible query formats, as well as more formal, i.e., structured, formats.

The goal of the IR system is to satisfy an information need with a generated answer. Most IR systems have a dataset of information, this dataset may be a corpus of text documents or the search index of a search engine. Thus, besides answering queries, an IR system also needs to maintain its dataset, i.e., extend it with new documents, remove faulty documents, and keep the data quickly accessible.

Additional information about the items in the dataset is also beneficial. A corpus of text documents does not only consist of sentences of text documents, but documents may also have meta data. Annotations and SCDs provide additional beneficial information for the IR system, too. Consequently, all additional information should be used to generate answers to queries.

Usually, a scoring function is used to calculate how well a item from the dataset matches the information need of the queries. Thus, the IR system calculates the score for all items in the dataset and returns the items ranked by the scores as answer.

In a sophisticated IR system, previous queries of the same and other users are considered, too. Such sophisticated systems lead us to agents (which follow in Section 2.3).

2.2.2. Information Extraction

IE is often used when processing text documents or in the scoring functions of IR systems. In the context of NLP, IE is about extracting structured information from natural language texts.

OpenIE [AJPM15] is a well-know approach to extract triples of subject, predicate, and object from sentences. These triples are often called *spo*-triples. For example, the sentence “Born in a small town, she took the midnight train going anywhere.” could result in the following triples of strings:

(she, took, midnight train), (she, took, train), (she, born in, small town)

An SCD for the sentences may contain these triples as additional data in the set \mathcal{C} . Furthermore, an IR system may use the *spo*-triples for calculating similarity scores of sentences based on equal *s*, *p*, and *o* values.

A big benefit of OpenIE is that it works out of the box and without training for English language texts. Thus, it can be used to automatically annotate a corpus with *spo*-triples.

2.2.3. Topic Models

Topic models represent the topics in a corpus. Commonly, a distribution of words is used to model each topic. Such distribution represents for each word in the corpus how probable the occurrence of each word in a document of the actual topic is. Thus, each topic is characterized by its most probable words.

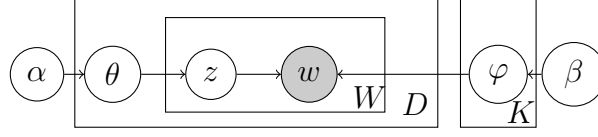


Figure 2.1.: Graphical representation of LDA in plate notation, inspired from [BNJ03]. Only the words of the document w are observable (marked grey).

A well-known technique to create topic models is LDA [BNJ03]. The algorithm LDA identifies for topics in the corpus during training. As the name states, it allocates latent topics using a Dirichlet distribution. An LDA topic model is trained on a corpus of text documents and it is assumed that each document is a set of words, also called bag-of-words. In the resulting model, each topic is described by a topic word distribution, i.e., a distribution of the most probable words, while each document is attributed with a document topic distribution, i.e., a distribution of the most probable topics. Hence, an LDA topic model describes clusters of similar documents sharing the same topics among our corpus.

Commonly, K is used for the number of topics a topic model shall create during training. With SCDs, K is used as number of SCDs in the SCD set $g(\mathcal{D})$ of a corpus. In both cases K describes a number of clusters of similar parts in corpus, i.e., document for LDA and sentences for SCDs. In general, SCDs can also be understood as a kind of topic model, albeit a different one than an LDA topic model.

An LDA training process starts with the compilation of the documents representing the corpus and a reasonable choice for the number of topics K . Choosing a good K is difficult without considering all documents of the corpus. A too large K leads to very blurred topics that are superimposed with other topics. A too small number of topics leads to very general topics with low selectivity to other topics. Thus, K must be determined anew for each application or corpus.

Additionally, there are two hyperparameters $\alpha, \beta \in \mathbb{R}^+$, whose choice determines the trade-off between the following two targets:

- (i) Match the words of each document to as few topics as possible (α).
- (ii) Choose as few relevant words as possible for each topic (β).

An LDA topic model consists of two trained discrete probability distributions. Both distributions are derived from the documents $d \in \mathcal{D}$ for the topics $k \in \{1, \dots, K\}$.

- The document topic distribution θ_d for each document d contains the probability with which an arbitrarily selected word in the document d belongs to topic k .

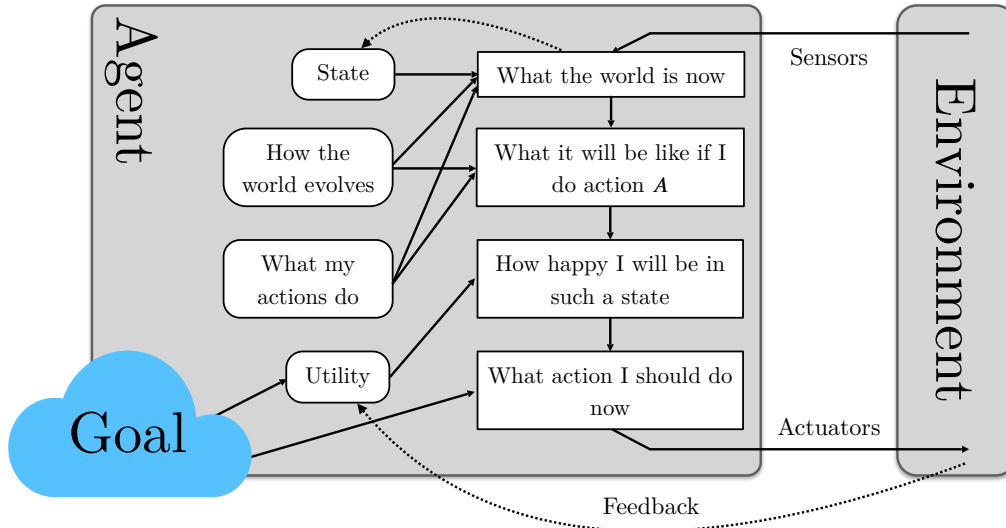


Figure 2.2.: An illustration of an agent, figure inspired by [RN21].

- The topic-word distribution φ_k for each topic k gives for an arbitrarily selected word from \mathcal{D} the probability the word belongs to topic k .

A graphical representation of LDA is shown in Figure 2.1. The hyperparameters α and β influence the distributions θ and φ . Each word w (of overall W words in the corpus) is assigned to a topic z .

An LDA topic model is a so-called generative model. A generative model in terms of LDA is characterized by the assumption that each document topic distribution generates actual words of each document.

We now know the recurring NPLs techniques used in this dissertation. Next, we describe the concept of intelligent agents.

2.3. Intelligent Agents

An agent [RN21] is a rational and autonomous unit acting in a world fulfilling a defined task. The world the agent exists in provides the environment. The agent is able to perceive this environment through sensors. Additionally, the agent is able to conduct actions through actuators and thus to change the environment. Generally, the available sensors and actions depend on the actual agent and its task. Figure 2.2 depicts a schematic illustration of an agent.

Internally, the agent is powered by decision, planning and learning processes, whereas some agents may only use a subset of these processes. The agent uses its sensors to

model the current environment in an internal state. Based on this state, the agent needs to decide which action to carry out next. Normally, the agent needs to select the *best* action from a set of possible actions. Such a selection requires knowledge about the consequences of each action and also of other influences the environment is exposed to, e.g., other agents acting in the same environment. To select the *best* action, the agent needs to calculate a score for each action considering the environment after conducting this action. This score represents the utility of each action, with the agent's goals in the background.

We assume that the goals of the agent represent an abstract task the agent has to fulfill. Based on the goals, the agent is able to plan a sequence of actions bringing it near the goals. Additionally, the agent may implement a feedback based learning process: For example, the principal, i.e., a user of the agent, gives feedback for each conducted action. This feedback may be incorporated by the agent to better fulfill the goals, i.e., the agent may update the utility function based on feedback.

In this dissertation, we create an IR agent based on SCDs. The agent works with user-supplied corpora, enriches corpora with SCDs, extends corpora with new text documents, provides an SCD-based document retrieval service, and incorporates feedback to further improve the SCDs.

The next chapter delves into SCDs and techniques for working with SCDs.

3. The Universe of Subjective Content Descriptions

SCDs provide a flexible way to add additional data to sequences of words organized in corpora of text documents. Generally, SCDs can contain any type of data and provide NLP techniques for understanding textual data. For example, SCDs can support agents by performing the following tasks: (i) Estimating SCDs for a single previously unseen text document using the Most Probably Suited SCD (MPS²CD) algorithm [KBBM19, KBBM20], (ii) classifying a text document as related, extended, revised, or unrelated to a corpus [KBBM20], (iii) moving the SCDs from one corpus to another similar corpus by adapting the SCDs’ domain [KBBM21], (iv) separating SCDs and actual content being interleaved in text documents [BBG⁺21b, BBG⁺21a], or (v) enriching SCDs in a corpus already sparsely associated with SCDs [KWM19].

We can look at SCDs from different perspectives. SCDs may represent the topics of a corpus and thus provide a topic model. But SCD can also simply provide a set of textual annotations for a corpus, similar to sticky notes in a book. In each case, SCDs represent a *subjective* view of a human user on the corpus and the user’s queries.

SCDs are based on some fundamental intentions and objectives that are common to all techniques for using SCDs. These fundamentals make SCDs valuable for IR agents. SCDs do not require large corpora nor huge datasets. Especially, when working with domain-specific texts or rare languages, the size of a corpus is limited and there is not enough data to train, e.g., an LLM. Though, SCDs provide a applicable technique in such case, e.g., for old Tamil poems [BBG⁺21a]. Furthermore, techniques using SCDs require less computational resources compared to LLMs and do not require special hardware like graphic cards. An SCD-based IR agent can run on off-the-shelf hardware, and thus can be considered as rather resource efficient. Summarized, SCDs represent a lean computing approach for text understanding.

This chapter starts with a formal introduction to SCDs including the supervised estimation of SCDs and MPS²CD. Afterwards, we describe how to create an SCD-based IR agent and outline problems in the field of SCDs. This includes previously solved problems and problems solved in this dissertation. Finally, we present the contribution of this dissertation in detail.

3.1. Subjective Content Descriptions

Kuhr et al. have introduced SCDs in [KBBM19]. SCDs provide additional location-specific data, e.g., annotations or sticky notes, for documents. The data provided by SCDs may be of various types, like additional definitions, labels or links to knowledge graphs or other SCDs. The theory of SCDs is not restricted to text documents annotated with additional text. However, in many use-cases we annotate text documents with additional textual definitions.

The SCD set $g(\mathcal{D})$ contains all SCDs including their referenced sentences and additional data. We assume that each SCD shows a specific distribution of words near the SCD's location, i.e., of the referenced sentence, in the document. Thus, each SCD forms an SCD-word distribution which models the words of its referenced sentences. All this distributions of the SCDs of a corpus are represented in an SCD-word distribution matrix [KBBM19], SCD matrix for short. The SCD matrix contains a row for each SCD and each row contains the word distribution from the referenced sentences of this SCD. We assume, that the SCD of a sentence generates the words of its referenced sentences. Thus, the SCD matrix can be interpreted as a generative model. Generally, a generative model for SCDs is characterized by the assumption that the SCDs generate the words of the documents in the corpus.

The SCD matrix $\delta(\mathcal{D})$ models the distributions of words for all SCDs $g(\mathcal{D})$ of a corpus \mathcal{D} and is structured as follows:

$$\delta(\mathcal{D}) = \begin{matrix} & w_1 & w_2 & w_3 & \cdots & w_L \\ \begin{matrix} t_1 \\ t_2 \\ \vdots \\ t_K \end{matrix} & \begin{pmatrix} v_{1,1} & v_{1,2} & v_{1,3} & \cdots & v_{1,L} \\ v_{2,1} & v_{2,2} & v_{2,3} & \cdots & v_{2,L} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{K,1} & v_{K,2} & v_{K,3} & \cdots & v_{K,L} \end{pmatrix} \end{matrix}$$

The SCD matrix consists of K rows, one for each SCD in $g(\mathcal{D})$, and each row contains the word probability distribution for the SCD. Therefore, the SCD matrix has L columns, one for each word in the vocabulary of the corresponding corpus. Thus, the word distribution of SCD t_i , the i -th row of SCD matrix $\delta(\mathcal{D})$, is the vector $(v_{i,1}, \dots, v_{i,L})$.

3.1.1. Supervised Estimation of SCDs

The Supervised Estimation of an SCD Matrix (SEM) is described in Algorithm 1. The input of SEM is a corpus \mathcal{D} and an SCD set $g(\mathcal{D})$, it returns the SCD matrix

Algorithm 1 Supervised Estimator of SCD Matrices $\delta(\mathcal{D})$

```

1: function SEM( $\mathcal{D}$ ,  $g(\mathcal{D})$ )
2:   Input: Corpus  $\mathcal{D}$ ; Set of SCDs  $g(\mathcal{D})$ 
3:   Output: SCD-word distribution matrix  $\delta(\mathcal{D})$ 
4:   Initialize an  $K \times L$  matrix  $\delta(\mathcal{D})$  with zeros
5:   for each document  $d \in \mathcal{D}$  do
6:     for each SCD  $t = (\mathcal{C}, \{s_1, \dots, s_S\}) \in g(d)$  do
7:       for  $j = 1, \dots, S$  do ▷ Iterate over sentences
8:         for each word  $w_i \in s_j$  do
9:            $\delta(\mathcal{D})[t][w_i] += I(w_i, s_j)$  ▷ Assume uniform, i.e.,  $I(w_i, s_j) = 1$ 
10:  return  $\delta(\mathcal{D})$ 

```

$\delta(\mathcal{D})$. SEM works in a supervised manner, because it requires the SCD set $g(\mathcal{D})$ containing referenced sentences.

SEM iterates over each document in the corpus and the document’s located SCDs. For each located SCD given t , the SCD matrix is updated. First, the referenced sentences $\{s_1, \dots, s_S\}$ of t are extracted. Next, for each sentence the row of the matrix representing SCD t gets incremented for each word in each referenced sentence s_j .

Kuhr et al. use a window moving over the words of the documents. Thus, they do not restrict SCDs to referenced sentences but in general to referenced windows of words. The authors assume an SCD generates the words in a certain radius around the SCD’s location while in this dissertation we assume an SCD generates the words of the sentence at the SCD’s location. Generally, each of our referenced sentences is a sequence of words and thus, our approach can be generalized to tumbling or sliding windows of words. However, a sliding window results in more computations and we argue that our sentence-wise approach does not result in a bad partitioning as in natural language a sentence is a logical unit. Thus, the most influential words of a word belong to the same sentence the word itself belongs to. The sentence-wise approach maintains the logical structure of the documents.

After Algorithm 1 has finished, the SCD matrix needs to be normalized row-wise to meet the requirements of a probability distribution. However, we skip the normalization because multiple calculations on small decimal values on a computer reduce the accuracy. Later, we use the cosine similarity with the rows of the matrix and the cosine similarity does a normalization by definition. Thus, by skipping the normalization we save computational resources and get slightly more accurate results in subsequent steps.

3. The Universe of Subjective Content Descriptions

Algorithm 2 Estimating MPS²CDs and similarity values

```

1: function MPS2CD( $d'$ ,  $\delta(\mathcal{D})$ )
2:   Input: Document  $d'$ ; SCD matrix  $\delta(\mathcal{D})$ 
3:   Output: SCDs  $g(d')$  with similarity values  $\mathcal{W}$ 
4:    $\mathcal{W} \leftarrow \emptyset$ 
5:    $g(d') \leftarrow \emptyset$ 
6:   for each sentence  $s_i^{d'} \in d'$  do
7:      $\delta(s_i^{d'}) \leftarrow$  new zero-vector of length  $L$ 
8:     for each word  $w \in s_i^{d'}$  do
9:        $\delta(s_i^{d'})[w] += I(w, s_i^{d'})$ 

10:    $t' \leftarrow \arg \max_{t \in g(\mathcal{D})} \frac{\delta(\mathcal{D})[t] \cdot \delta(s_i^{d'})}{\|\delta(\mathcal{D})[t]\|_2 \cdot \|\delta(s_i^{d'})\|_2}$ 
11:    $sim \leftarrow \max_{t \in g(\mathcal{D})} \frac{\delta(\mathcal{D})[t] \cdot \delta(s_i^{d'})}{\|\delta(\mathcal{D})[t]\|_2 \cdot \|\delta(s_i^{d'})\|_2}$ 
12:    $g(d') \leftarrow g(d') \cup t'$  ▷ Also add  $s_i^{d'}$  as referenced sentence to  $t'$ 
13:    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(t', sim)\}$ 
14: return  $g(d')$ ,  $\mathcal{W}$ 

```

3.1.2. Most Probably Suited Subjective Content Descriptions

The previously described and trained SCD matrix can be used to estimate SCDs for a document without associated SCDs. First we formalize the MPS²CD problem and afterwards solve the problem by Algorithm 2 using the SCD matrix [KBBM19].

The MPS²CD problem asks for the k' most probably suited SCDs $t_1, \dots, t_{k'}$ for a document d' given the SCD matrix $\delta(\mathcal{D})$:

$$g(d') = \arg \max_{t_1, \dots, t_{k'} \in g(\mathcal{D})} P(t_1, \dots, t_{k'} \mid d', \delta(\mathcal{D}))$$

The definition of the MPS²CD problem does not consider the sentence-wise iteration used while training the SCD matrix. We can reformulate the MPS²CD problem to consider the sentence-wise iteration:

$$g(d') = \bigcup_{\substack{\text{sentences} \\ s_i^{d'} \in (s_1^{d'}, \dots, s_{M^{d'}}^{d'})}} \arg \max_{t \in g(\mathcal{D})} P(t \mid s_i^{d'}, \delta(\mathcal{D}))$$

Analogous to the second definition of the MPS²CD problem, Algorithm 2 iterates over each sentence of d' . For each sentence the algorithm creates the word distribution vector $\delta(s_i^{d'})$. This vector of the word counts in sentence $s_i^{d'}$ is shaped $1 \times L$ and

is created using the approach that was used for the rows of the matrix in Algorithm 1. In addition to $\delta(s)$, \vec{s} also represents the word counts vector of a sentence s .

We use the cosine similarity to compare $\delta(s_i^{d'})$ with a row of the SCD matrix $\delta(\mathcal{D})[t]$ representing SCD t :

$$\frac{\delta(\mathcal{D})[t] \cdot \delta(s_i^{d'})}{\|\delta(\mathcal{D})[t]\|_2 \cdot \|\delta(s_i^{d'})\|_2}$$

The product of Euclidean norms in the denominator is for normalization, so the vectors do not need to be normalized beforehand. The most probably suited SCD t is defined as the SCD belonging to the row resulting in the highest cosine similarity value.

The MPS²CD algorithm allows us to estimate the most probably suited SCDs for any sentence given the words of the sentence and the SCD matrix. Thus, using MPS²CD we are able to annotate a new and unseen text document d' with suitable SCDs from $g(\mathcal{D})$. However, we still need an initial set of SCDs for an initial corpus \mathcal{D} . Next, we consider how to integrate SCDs in an IR agent.

3.2. SCDs in an Information Retrieval Agent

We assume an IR agent works with a corpus of text documents. In this dissertation, the text documents are associated with SCDs. In Figure 3.1 the basic structure of an SCD-based IR agent is shown. Creating a fully functional SCD-based IR agent poses some problems to be solved in this dissertation.

3.2.1. Problem I: No SCDs Available

Let us consider the following abstract use-case: A user, e.g., a human, (bottom right corner of figure) brings a corpus of text documents to work with. This corpus represents the general field of interest for the user. The user might as well be an agent. Then, the user will send queries about the corpus to the IR agent, and the IR agent could answer these queries using SCDs if they are derived. To do so, the IR agent needs SCDs and an SCD matrix for the user supplied corpus. Obtaining SCDs is Problem I, because SEM requires an initial set of SCDs for a corpus, but a user supplied corpus often does not have SCDs. The lightning symbol in the top left corner of Figure 3.1 represents Problem I.

In our understanding, the initial set of SCDs used by SEM provides the supervision of the estimation of SCDs. Thus, to solve Problem I, an unsupervised technique

3. The Universe of Subjective Content Descriptions

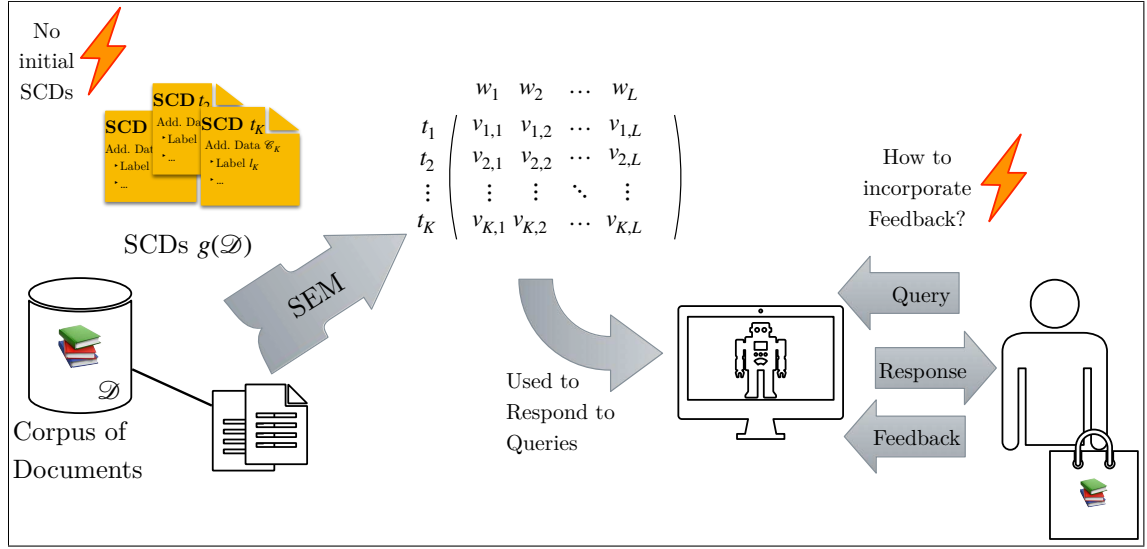


Figure 3.1.: Basic structure of an SCD-based IR agent. The lightning symbols show previously open problems solved in this dissertation.

for estimating SCDs is required. There are already some techniques introduced by Kuhr et al. which reduce the required supervision for estimating SCDs.

The context-specific adaptation of SCDs [KBBM21] provides a technique to transfer SCD matrices and their SCDs from one corpus to another corpus. The other corpus does not need any SCDs in advance, but both corpora must be similar in terms of having a similar vocabulary and covering a similar topic.

Another approach is to enrich SCDs in a corpus [KWM19]. Given a corpus with very few SCDs, the corpus gets enriched with further similar SCDs automatically, but some initial SCDs are required.

In some cases, documents are already annotated with SCDs, but there are SCDs embedded in the documents. Thus, the agent first needs to separate the embedded SCDs from the *content* in the documents. The inline SCD (iSCD) problem addresses this task [BBG⁺21b, BBG⁺21a]. Though, the solution still needs an SCD matrix.

Example 3.1. Inline SCD Example

Assume a document contains the following sentence with two SCDs embedded. The underlined words represent the SCDs, while the other words form the content.

“We visited the bisons large animals in the zoo a place where non-domestic animals are exhibited.”

In summary, prior to this dissertation, there is no technique for estimating SCDs for any corpus that requires no initial SCDs at all.

3.2.2. Answer Queries

If the IR agent has a corpus associated with SCDs and the SCD matrix, it can answer queries. We assume the query consists of one or multiple sentences to which similar documents should be retrieved from the corpus. First, the MPS²CD algorithm calculates the MPS²CDs for the query sentences. Second, for each MPS²CD the referenced sentences are fetched. Thus, the SCDs with their referenced sentences provide the response, whereas each sentence highlights a relevant part of a document. The MPS²CD similarity values can be used as a score and to order the SCDs in the response.

The quality of the response also depends on the text documents available in the corpus. If the agent does not have a relevant document it cannot retrieve one. Thus, the agent is required to maintain the corpus and extend it with relevant documents. Kuhr et al. [KBBM20] present an SCD-based classification approach for documents. New documents are classified as related, extended, revised, or unrelated to a corpus. Using this classification approach, the IR agent is able to identify relevant documents to add to the corpus. Such documents might be extensions and revisions because they contain possibly new changes of known documents. The new documents can be associated with SCDs using MPS²CD.

3.2.3. Problem II: Incorporate Feedback

After answering a query of a user, the IR agent may get feedback about the response. This leads us to the Problem II building an SCD-based IR agent. The agent should incorporate the feedback and the SCDs based on the feedback, but there is no algorithm to update SCDs matrices. The right lightning symbol of Figure 3.1 represents Problem II.

3.2.4. Problem III: Suitable Labels for SCDs

The response of the IR agent contains MPS²CDs for the query sentence of the user. Each SCD has a set of referenced sentences, which are part of a document. However, it is difficult for the IR agent to describe the response to the user such that the user can grasp each SCD. Especially, the response is only a set of SCDs with a set of

3. The Universe of Subjective Content Descriptions

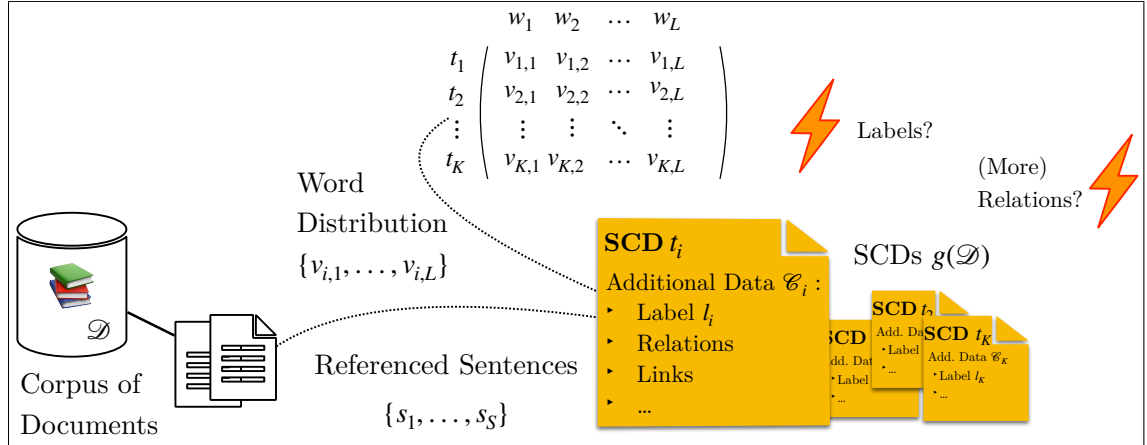


Figure 3.2.: Internal view on the SCDs used by the SCD-based IR agent. The lightning symbols show previously open problems solved in this dissertation.

sentences each. Thus, the user needs to read multiple sentences of each SCD and then select the SCDs which fit the user's information need best.

A label or short description of each SCD would solve Problem III. The IR agent can respond with a set of SCDs and their labels. Then, the user reads the labels of each SCD and selects the best one.

Figure 3.2 shows an internal view of the SCDs used by the IR agent. Each SCD, depicted as a sticky note, refers to a set of sentences in the corpus and a word distribution stored in the SCD matrix. In the figure, the additional data of the SCD is written directly on the the sticky note. There, a label l_i is already added, but for some SCDs it needs to be automatically estimated first. So the lightning symbol in the middle right of Figure 3.2 represents the Problem III.

3.2.5. Problem IV: Relations among SCDs

An SCD covers a group of similar sentences in the corpus. A sentence may be in an SCD or not, but in reality there is more than just similarity and boolean *in or out*. Thus, the additional content of an SCD may contain relations to other SCDs and an SCD may be linked to other SCDs.

Problem IV, outlined by the right lightning symbol in Figure 3.2, is about representing different relations about SCDs. Actually, different relations among SCDs may be a useful extra feature of SCDs rather than a problem. However, adding relations among SCDs requires changes to the SCD matrix, SEM, and the MPS²CD algorithm.

Creating an IR agent based on SCDs we came across four problems, (Problem I) most corpora initially do not contain any SCDs, (Problem II) feedback from the user cannot be used to improve the SCDs, (Problem III) SCDs need labels when shown to a user, and (Problem IV) grouping similar sentences is sometimes not enough, i.e., different relations among SCDs are needed.

3.3. Detailed Contributions

This dissertation solves the four problems identified in the previous section. Each chapter of Part I addresses one problem and provides a solution. In Part II, we build the described IR agent system as a web based information system and describe further applications of SCDs. Next, a brief preview of each chapter is provided.

3.3.1. Unsupervised Techniques

The first two chapters of Part I address the unsupervised estimation of SCDs and short descriptions or labels.

USEM UnSupervised Estimation of SCD Matrices (Chapter 4)

A user supplies a corpus to our IR agent and needs to retrieve documents with similar content and highlight relevant locations in retrieved documents. The IR agent needs SCDs referencing sentences of similar content across various documents in the corpus and most text documents are not associated with SCDs. We present USEM which solves the problem because it associates any corpus with SCDs. In an evaluation, we show that the performance of USEM in estimating topics of similar content in the corpus is on par with LDA, while USEM provides SCDs referencing sentences of similar content.

USEM solves Problem I, shown by the top left lightning symbol in Figure 3.1.

LESS Label Estimation for SCDs without Supervision (Chapter 5)

SCDs estimated by USEM lack meaningful descriptions, i.e., labels consisting of short summaries. Labels are important to identify relevant SCDs and documents by the agent and its users. We present LESS, an algorithm which creates labels for SCDs using the word distributions of the SCDs. In an evaluation, we compare the labels computed by LESS with labels computed by LLMs and show that LESS computes similar labels but requires less data and computational power.

LESS solves Problem III, shown by the middle right lightning in Figure 3.2.

3.3.2. Feedback and Updates

The next two chapters of Part I deal with incorporating feedback to improve SCDs, i.e., Problem II, which is outlined by the right lightning symbol in Figure 3.1.

FrESH Feedback-reliant Enhancement of SCDs by Humans (Chapter 6)

A human interacts with our IR agent and a response contains an erroneous part. Such errors, like faulty SCDs, should be sent back to the agent by the human as feedback. Then, the agent needs to incorporate the feedback and remove the erroneous part of its internally used SCDs. However, removing a faulty sentence with an SCD in a previously trained model is a difficult task—often the model needs to be retrained from scratch. To circumvent retraining, we present FrESH to keep the SCDs fresh and maintained with human feedback.

ReFrESH Relation-preserving Feedback-reliant Enhancement of SCDs (Chapter 7)

Again, we work with human feedback sent back to the agent after a response. In difference to FrESH, we do not have a faulty sentence or SCD to remove completely. We are in the situation, when a user of the agent is not the creator of the SCDs for the corpus. Hence, answers may be considered faulty by an agent’s user, because the SCDs may not exactly match the perceptions of an agent’s user. A naive and very costly approach would be to ask each user to completely create all the SCD themselves. To circumvent manual creation, we present ReFrESH, which updates the SCDs in the corpus incrementally. Using ReFrESH, SCDs can be refreshed with feedback by humans and it allows users to build even better SCDs for their needs.

3.3.3. Relations and Complementarity

The final chapter of Part I (Chapter 8) deals with adding relations among SCDs using the example of complementarity, i.e., Problem IV, which is outlined by the right lightning symbol in Figure 3.2.

Our agent so far relies on similarity measures to identify related documents used as response or for corpus extension. However, similarity may not be appropriate if looking for new information or different aspects of the same content. Therefore, we combine complementarity- and similarity-based identification of documents, specifically, (i) a formal definition of complementarity using the available SCDs in the form of relational tuples as well as a taxonomy interrelating the concepts of the tuples, (ii) a technique for classifying complementary and related documents in one go, and (iii) a case study assessing the classification performance for complementary and related documents.

We introduce a combined SCD (cSCD) matrix that is able to model similar and complementary SCDs altogether. This cSCD matrix comes with adaptations for SEM and MPS²CD which can be generalized for other relations than complementarity.

3.3.4. Application

There are two chapters in Part II. We present three applications of SCDs and thus demonstrate the usability of SCDs in further use-cases.

SIS SCD-based Information System (Chapter 9)

We present an implementation of the SCD-based IR agent described in Section 3.2. The agent is integrated in an information system to be accessed by humans via a Graphical User Interface (GUI) and by other applications via an Application Programming Interface (API). Users of the system may submit their own corpora and run queries. The system applies the theoretical contributions from Part I of this dissertation. We describe the basic structure and demonstrate how the system can assist a user.

ChatHA Humanities Aligned Chatbot (Chapter 10, Section 10.2)

We present a use-case of SCDs. ChatHA provides a user-friendly interaction of humans with their corpora. Internally, ChatHA relies on a chatbot with an LLM having access to the user’s corpus. The user can send queries about the corpus to the LLM and the LLM generates answers. SCDs are used to eliminate hallucinations in the raw LLM output. Further, SCDs can be used to add citations to the LLM’s output pointing to the user’s corpus.

RDR Integration Research Data Repository (Chapter 10, Section 10.3)

We outline how SCDs can be used to rapidly deploy an information system for a corpus of text documents. RDRs are web based information system containing large amounts of data in multiple collections. Often there are collections of text documents which represent a corpus. A user interested in viewing a corpus normally needs to download and manually search the corpus. An automatic import in our SCD-based information system providing an IR agent makes it easier for such a user to interact with a corpus in the RDR.

Part I.

Theoretical Foundation

4. USEM: UnSupervised Estimation of SCDs

This chapter is based on the following two publications:

- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **Unsupervised Estimation of Subjective Content Descriptions in an Information System** in *International Journal of Semantic Computing*, 2024
<https://dx.doi.org/10.1142/S1793351X24410034>
- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **Unsupervised Estimation of Subjective Content Descriptions** in *17th IEEE International Conference on Semantic Computing (ICSC 2023)*
<https://dx.doi.org/10.1109/ICSC56153.2023.00052>

In both cases: Magnus Bender developed the initial idea, conducted the experiments, and wrote the manuscript. The other three authors fundamentally supervised the research by discussing ideas, proofreading the manuscript multiple times, and giving feedback. The actual sections of this chapter are largely taken verbatim from the journal article. The journal article itself is an extended version of the conference paper.

4.1. Introduction

In this chapter, we present the solution to Problem I (Subsection 3.2.1) we came across building the SCD-based IR agent. USEM is the UnSupervised Estimator for SCD Matrices, an approach to estimate an SCD matrix for any corpus in an unsupervised manner.

We assume that a user provides a corpus for IR. However, this corpus is not associated with SCDs, which are required for IR. SCDs are subjective in the sense that the agent associates data with text parts, e.g., sentences, to best carry out tasks described by the principal. Thus, it is crucial for the IR agent to add corpus-specific SCDs to the user supplied corpus.

For the IR agent, it is valuable to have for each sentence a set of references to similar sentences in the documents across its corpus. Using these references, the agent can

retrieve sets of similar sentences in response to each sentence a user queries. Such references can be modeled by SCDs being associated with the sentences, including the corresponding SCD matrix. In our understanding, each SCD represents a concept or topic mentioned in the corpus. Each SCD’s concept is implicitly defined by the word distribution and the content of the sentences referenced by each SCD. Thus, the sentences of an SCD are similar.

Let us have a closer look why sentence similarity can help with the IR task. Given is a corpus of three documents $\{d_1, d_2, d_3\}$, dealing with three different car models, and each document consists of ten sentences $\{d_i = (s_1^{d_i}, s_2^{d_i}, \dots, s_{10}^{d_i})\}_{i=1}^3$. For the agent answering a query about sentence $s_2^{d_2}$, the SCDs t_1 and t_2 could be valuable: SCD t_1 references the sentences $s_2^{d_2}$ and $s_8^{d_1}$ because both sentences are about the engine’s horsepower. Thus, t_1 represents the concept *engine power*. Furthermore, SCD t_2 represents the concept *car manufacturer* because it references two sentences $s_7^{d_3}$ and $s_2^{d_1}$ about the car’s manufacturer. Then, the agent returns d_1 and highlights $s_8^{d_1}$ answering the query about sentence $s_2^{d_2}$ because both sentences cover *engine power*. So for this query, the additional information of t_1 turned out to be useful.

In a first step, a solution to the lack of SCDs for most corpora identifies similar sentences—preferably in an unsupervised manner. Then, using the identified similar sentences, SCDs are formed, where each SCD represents a different concept in the corpus and references multiple locations in the text documents of the corpus. This is how USEM associates any corpus of text documents with SCDs. Mainly, USEM detects similar concepts referenced in the text documents of the corpus and then forms an SCD, which groups all occurrences of the same concept.

The remainder of this chapter is structured as follows: First, we conclude the introduction by describing the analogies between SCD matrices and topic models. Afterwards, we then describe the problem of estimating SCDs in an unsupervised manner and our solution USEM. We introduce three methods for USEM, namely a greedy, a K-Means [Llo82], and a DBSCAN [EKSX96] based method. An IR agent has to automatically select one best method and the best hyperparameters. Thus, we provide a model selection approach for SCD matrices. In the end, we compare USEM with its three methods in an evaluation against the well-know LDA. Finally, we look at related work and provide an interim summary afterwards.

Generally, each estimated SCD represents a topic of the corpus, which is why an SCD matrix can be interpreted as a topic model of the corpus. In contrast to LDA, USEM associates each sentence in the corpus with an SCD, while LDA associates each single word with a topic and each text document with a topic distribution. An SCD consists of multiple referenced sentences in the corpus and the sentences’ overall word distribution, while LDA’s topics consist of a distribution of words associated with each topic. Hence, associating SCDs with sentences instead of words or text documents is the important difference.

4.2. Unsupervised Estimation of SCDs

The unsupervised estimation of SCDs is divided into two steps. In a first step, an SCD matrix needs to be estimated for a corpus. Given an estimated SCD matrix, the SCDs of a corpus are defined by their SCD-word distributions and the referenced sentences. For USEM, however, there are three methods with multiple hyperparameters resulting in multiple estimated SCD matrices for each corpus. Thus, in a second step, an IR agent needs to select one SCD matrix, for which we introduce a model selection approach.

4.2.1. Unsupervised Estimation of SCD Matrices

This subsection introduces USEM, the UnSupervised Estimator for SCD Matrices. The SCD matrix represents in its rows each SCD found in the corpus. Each row contains the word distribution of the sentences associated with the row's SCD. USEM is also a kind of topic estimation algorithm because each SCD represents a concept in the corpus and the SCD references the sentences dealing about this concept.

Algorithm 3 outlines USEM. The input of USEM is a corpus for which an SCD matrix is to be computed. Commonly, a sentence is associated with an SCD and each SCD references one or multiple sentences. USEM initially starts by associating each sentence to one unique SCD. The SCD's word distribution of each SCD then only contains the words of the referenced sentence. Lines 10 - 14 of Algorithm 3 show how to create this initial SCD matrix, which consists of a row for each sentence in the document's corpus. The word distributions are calculated using the influence value the same way as in Algorithm 1 (SEM).

The next step is to find the sentences that represent the same concept and group them into one SCD. There are three different methods for detecting similar rows in the initial SCD matrix. Lines 16 - 33 of Algorithm 3 show the three methods and how the rows are merged. If there are more than two rows, two are merged at a time until all are merged. The main idea of merging two rows is to sum up the quantities of each word in both distributions of words and deleting the second row from the matrix.

To identify similar sentences, USEM has three different methods. The first is a greedy approach followed by two well-known clustering techniques, K-Means and DBSCAN. We use DBSCAN and K-Means because each method represents a clustering method following a different approach, i.e., density based and distance based clustering.

Algorithm 3 UnSupervised Estimator for SCD Matrices $\delta(\mathcal{D})$

```

1: function USEM( $\mathcal{D}$ ,  $m$ ,  $[\theta]$ ,  $[K]$ ,  $[\varepsilon]$ ,  $ms$ )
2:   Input: Corpus  $\mathcal{D}$ ; Method with hyperparameters, i.e.,
3:      $m = \text{Greedy}$  and threshold  $\theta$ ,
4:      $m = \text{K-Means}$  and number of SCDs  $K$ , or
5:      $m = \text{DBSCAN}$ , distance  $\varepsilon$ , and threshold  $ms$ 
6:   Output: SCD-word distribution matrix  $\delta(\mathcal{D})$ 
7:   Initialize an  $M \times L$  matrix  $\delta(\mathcal{D})$  with zeros
8:    $l \leftarrow 0$ 
9:   ▷ Build initial SCD matrix
10:  for each document  $d \in \mathcal{D}$  do
11:    for each sentence  $s^d \in d$  do
12:      for each word  $w_i \in s^d$  do
13:         $\delta(\mathcal{D})[l][w_i] += I(w_i, s^d)$ 
14:       $l \leftarrow l + 1$ 
15:      ▷ Use method  $m$  to merge rows
16:  if  $m = \text{Greedy}$  then
17:    repeat ▷ Detect similar rows and merge
18:       $(r_i, r_j) \leftarrow \text{MOSTSIMILARROWS}(\delta(\mathcal{D}))$ 
19:       $\delta(\mathcal{D})[r_i] \leftarrow \delta(\mathcal{D})[r_i] + \delta(\mathcal{D})[r_j]$  ▷ Sum rows
20:       $\delta(\mathcal{D})[r_j] \leftarrow \text{Nil}$  ▷ Delete row
21:    until  $\text{SIMILARITY}(r_i, r_j) < \theta$ 
22:  else ▷ Create clusters of similar rows
23:    if  $m = \text{K-Means}$  then
24:       $clusters \leftarrow \text{KMEANS}(\delta(\mathcal{D}), K)$ 
25:    else
26:       $clusters \leftarrow \text{DBSCAN}(\delta(\mathcal{D}), \varepsilon, ms)$ 
27:    for each cluster  $c \in clusters$  do
28:      ▷ Create sum of all cluster's rows in first row
29:       $r_i \leftarrow \text{FIRSTROW}(c)$ 
30:       $\delta(\mathcal{D})[r_i] \leftarrow \sum_{r_j \in c} \delta(\mathcal{D})[r_j]$ 
31:      for each row  $r_j \in c$  do
32:        if  $r_i \neq r_j$  then ▷ Delete all non-first rows
33:           $\delta(\mathcal{D})[r_j] \leftarrow \text{Nil}$ 
34:  return  $\delta(\mathcal{D})$ 

```

Greedy by Similarity The first method greedily selects the next two rows to merge. It calculates the cosine similarity between all rows, containing the word distributions, in the matrix and always merges the two most similar rows. This is repeated until the similarity between the two most similar rows is below the threshold θ (Algorithm 3, Lines 17 - 21). Thus, with a lower threshold less SCDs with more referenced sentences each will be estimated and a higher threshold leads to more SCDs with less referenced sentences.

The calculation of the cosine similarity between all rows is realized as a matrix multiplication:

$$S_{\delta(\mathcal{D})} = \frac{\delta(\mathcal{D}) \cdot \delta(\mathcal{D})^T}{\|\delta(\mathcal{D})\|_2 \cdot \|\delta(\mathcal{D})\|_2^T}$$

The numerator represents the dot product between each row of the matrix to each other and the denominator contains the lengths of each row to normalize the matrix's rows, as $\|v\|_2$ represents a vector of the Euclidean norm of each row in v and the symbol \cdot the matrix multiplication. Numerator and denominator are matrices of size $K \times K$ each, which are then divided element-wise to form the cosine similarity matrix $S_{\delta(\mathcal{D})}$. After doing so, $S_{\delta(\mathcal{D})}$ contains the cosine similarity between each pair of rows in the matrix $\delta(\mathcal{D})$. The two most similar rows in $\delta(\mathcal{D})$ can now be identified by searching for the highest value in $S_{\delta(\mathcal{D})}$, of course without searching the diagonal. Row and column index of the highest value in $S_{\delta(\mathcal{D})}$ represent the most similar rows in $\delta(\mathcal{D})$. Both indexes are returned by MOSTSIMILARROWS in Algorithm 3.

Matrix multiplications on huge matrices can be computationally expensive. In case of the SCD matrix, it is a sparse matrix and sparse matrix multiplication is reasonably fast. Additionally, the Euclidean norms of the rows can be cached and updated partially for the changed rows, only.

K-Means One well-know clustering technique is K-Means [Llo82]. We will not get into the details how K-Means works, but focus on how to apply K-Means. K-Means is initialized with K centroids of which each centroid represents a cluster. Each point is assigned the nearest centroid in terms of the Euclidean distance using a vector representation of the point. Iteratively, the clusters are optimized by aligning each centroid in the center of all the points contained in the centroid's cluster.

We run K-Means on the rows of the SCD matrix to detect clusters of similar rows in the initial SCD matrix. Each row represents a point and the word distribution is the vector representation of this point. After K-Means is finished, USEM merges the rows of the matrix included in the same cluster (Algorithm 3, Lines 27 - 33). Hence, the number of clusters is equal to the number of SCDs in the end. As hyperparameter, the number of SCDs to estimate K is specified. Alternatively, K can be specified by a factor to multiply with the initial number of sentences in the

corpus, e.g., the factor 0.25 sets the number of SCDs to a quarter of the sentences in the corpus. Furthermore, there are techniques to estimate a *good* number of cluster for K-Means on the corpus [PM⁺00].

DBSCAN Another well-know clustering technique is DBSCAN [EKSX96]. In contrast to K-Means, DBSCAN is able to detect concave structures in data and works in a density based way. DBSCAN clusters two points together if both are in a neighborhood, the distance making up a neighborhood is defined by the hyperparameter ϵ . A cluster then grows by adding all points in the neighborhood to the same cluster. Additionally, there is a minimum samples threshold ms which defines the minimum number of points needed to form a cluster.

We run DBSCAN on the cosine similarity matrix $S_{\delta(\mathcal{D})}$ and again merge the rows of the matrix included in the same cluster (Algorithm 3, Lines 27 - 33).

Comparing the three methods, when using K-Means the number of SCDs to estimate K has to be specified in beforehand. The greedy method and DBSCAN determine the number of SCDs on their own. Though, the greedy method needs a similarity threshold θ and DBSCAN ϵ and the minimum samples threshold ms .

We cannot predict which method works better for a given corpus. As typical for greedy methods, we expect the greedy method working well for higher thresholds and more SCDs to estimate, while for smaller thresholds and a small number of SCDs, the greedy method will miss the global optimum.

4.2.2. Model Selection for SCD Matrices

This subsection introduces a model selection approach for SCD matrices to automatically select the best method with the best hyperparameters for USEM.

First, we have to determine what a good SCD matrix is and define a quality score to represent the quality of an SCD matrix estimated by USEM. This score needs to be calculated based on the estimated SCD matrix. Hence, possible input values are the word distributions and the referenced sentences for each SCD. However, there is no supervision and we do not have any ground truth to validate the SCDs against. Thus, we have to use quantitative attributes of the estimated SCD matrices.

Each SCD references a number of sentences and we can use these numbers of references to measure the quality of a matrix. We argue that a good SCD references a smaller amount of sentences, i.e., a reference to 100 or more similar sentences in the corpus is less beneficial for a human working with the corpus than a reference to fewer sentences. Additionally, an SCD referencing only one or two sentences is not

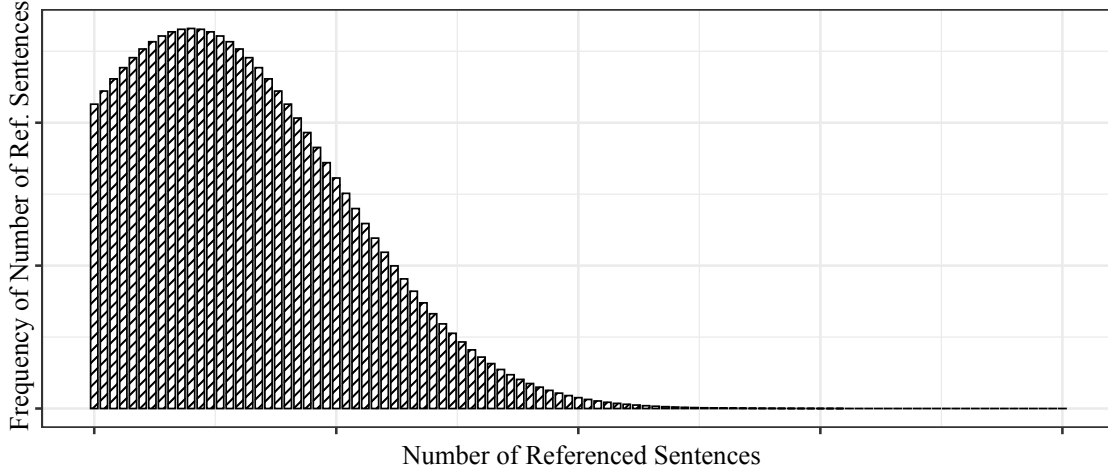


Figure 4.1.: Desired optimal distribution of the number of referenced sentences for an SCD matrix. A histogram depicting the different numbers of sentences referenced in an SCD matrix should show a similar course.

really beneficial either. Based on these deliberations, we assume that the distribution of the number of referenced sentences for an SCD matrix shown in Figure 4.1 is optimal. Similar to a histogram, on the x-axis the number of referenced sentences for the SCDs is displayed and on the y-axis the desired frequencies of each number of sentences are displayed. We omit the values on the axes because the actual values are not of relevance here—the course of the graph is the crucial point. For the sake of completeness: The graph shows a discretized normal distribution with mean 10 and a standard derivation of 15 in the interval from 0 to 100.

Returning to the quality score of a matrix, we need a possibility to compare the distribution of different numbers of sentences referenced in an SCD matrix with the assumed optimal normal distribution. Therefore, we scale the distribution of different numbers of sentences to the interval from 0 to 100. Furthermore, we discretize the normal distribution. Afterwards, the distance of both distributions can be calculated with the Hellinger distance [Hel09]

$$\text{HD}(u, v) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^k (\sqrt{u_i} - \sqrt{v_i})^2},$$

where u and v represent a vector of each distribution. By calculating $1 - \text{HD}(u, v)$ this distance is converted to a similarity score representing an SCD matrix' quality. Given this quality score based on the Hellinger distance, we now can select the best SCD matrix given a set of matrices trained by USEM.

Algorithm 4 SCD Matrix Model Selection

```

1: function ESTIMATEBESTMATRIX( $\mathcal{D}$ )
2:   Input: Corpus  $\mathcal{D}$ 
3:   Output: Best SCD-word distribution matrix  $\delta(\mathcal{D})$ 
4:    $sim_{best} \leftarrow 0$ 
5:    $\delta_{best} \leftarrow Nil$  ▷ Iterate all methods
6:   for each method  $m \in \{\text{Greedy, K-Means, DBSCAN}\}$  do
7:     ▷ Take a set of hyperparameters depending on method
8:     if  $m = \text{Greedy}$  then
9:        $H \leftarrow (0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1)$  ▷ Values of  $\theta$ 
10:    if  $m = \text{K-Means}$  then
11:       $M \leftarrow \sum_{d \in \mathcal{D}} M^d$  ▷ Number of sentences in  $\mathcal{D}$  to calculate  $K$  values
12:       $H \leftarrow (\lfloor M \cdot 0.8 \rfloor, \lfloor M \cdot 0.6 \rfloor, \lfloor M \cdot 0.4 \rfloor, \lfloor M \cdot 0.3 \rfloor, \lfloor M \cdot 0.2 \rfloor, \lfloor M \cdot 0.1 \rfloor)$ 
13:    else
14:       $H \leftarrow ((0.3, 1), (0.5, 10), (0.5, 5), (0.5, 2), (0.7, 10))$  ▷ Tuples of  $\varepsilon, ms$ 
15:    for each hyperparameter  $h \in H$  do
16:       $\delta(\mathcal{D}) \leftarrow \text{USEM}(\mathcal{D}, m, h)$  ▷ Run USEM
17:      ▷ Calculate score using Hellinger distance to normal distribution
18:       $sim \leftarrow 1 - \text{HD}(\text{SCALE}([0, 100], \delta(\mathcal{D})), \mathcal{N}([0, 100], \mu = 10, \sigma^2 = 15))$ 
19:      if  $sim > sim_{best}$  then
20:         $sim_{best} \leftarrow sim$ 
21:         $\delta_{best} \leftarrow \delta(\mathcal{D})$ 
22:  return  $\delta_{best}$ 

```

The entire model selection approach is described by Algorithm 4. The algorithm takes a corpus and returns the best SCD matrix. In Line 6 it starts with iterating over all three methods of USEM. Depending on the method in Lines 8 - 14 different sequences of hyperparameters H to try are chosen. These hyperparameters shall cover a wider range of possibly good hyperparameters for the corpus. Through the for loop starting in Line 15 for each method and each previously chosen hyperparameter in H USEM estimates an SCD matrix for corpus \mathcal{D} . Afterwards, in Line 18 the resulting SCD matrix is scored using the Hellinger distance after scaling the result as described above. Finally, Line 22 returns the SCD matrix resulting in the highest quality score.

So far, we have introduced USEM including a model selection approach for the resulting SCD matrices. Next, we describe and discuss the workflow, dataset, and implementation used in our evaluation along with the results comparing USEM against LDA.

4.3. Evaluation

After we have introduced USEM with its three methods, we present an evaluation. First, we describe the used corpus and evaluation metrics. Finally, we present the results of the evaluation and demonstrate the performance of USEM in comparison to LDA.

4.3.1. Dataset

In this evaluation we use the Bürgerliches Gesetzbuch (BGB)¹, the Civil Code of Germany, in German language as corpus. Therefore, OpenIE [AJPM15] cannot be used and thus the BGB is an example where we could benefit from USEM. The BGB is freely available and can be downloaded as an XML file. Therefore, it is easily parsable and processable. As the corpus is a law text it consists of correct language, i.e., punctuation and spelling follow the orthographic rules. Thus, less preprocessing and no data cleaning is needed. Furthermore, the words used in text documents have a clear meaning and mostly the same words are used instead of using synonyms.

The entire corpus consists of 2 462 law paragraphs and overall 8 020 sentences which are used as SCD windows. Each law paragraph contains between 1 and 45 sentences

¹<https://www.gesetze-im-internet.de/bgb/>, English translation https://www.gesetze-im-internet.de/englisch_bgb/

with an average of 3.3 sentences. The vocabulary consist of 5 294 words, where each sentence is between 1 and 51 with an average of 10.9 words long.

4.3.2. Metrics

Topic models are trained in an unsupervised way using statistical methods, thus, the topics gained by LDA are statistically optimized but may not match human judgement of *good* topics. In general, automatically evaluating the quality of a model from a human point of view is a difficult task. A common measure to evaluate the interpretability of topics regarding human judgement is coherence. Röder et al. [RBH15] compare and evaluate multiple coherence measures against human judgement as gold standard. The authors gain the best results using the C_V measure. However, due to negative correlations and problems reproducing the C_V values in their paper, Röder does not recommended to use the C_V coherence any more². Therefore, in our evaluation we use the UMass coherence calculated using Gensim’s coherence model.

As already stated in Subsection 4.2.2, the number of referenced SCD windows per SCD is relevant. For example, having 1 000 SCD windows and 100 SCDs, each SCD should have a similar number around 10 referenced SCD windows. It would be bad, if 99 SCDs reference 1 window each and the 1 remaining SCD references the remaining 901 windows. Therefore, we evaluate the number of referenced windows per SCD. Besides showing all numbers of referenced windows, we also show the numbers only for SCDs with two or more referenced windows, i.e., we interpret SCDs with only one referenced window as an irrelevant SCD and omit those SCDs.

For LDA an evaluation of referenced documents per topic is not necessary, as the training ensures a similar number of referenced topics per document.

4.3.3. Workflow and Implementation

USEM is implemented using Python and runs inside a Docker container. The implementation uses the libraries Gensim³, NumPy⁴, and NLTK⁵. The evaluation of USEM follows this workflow:

²“The usage of the C_V coherence is not recommended anymore!”, stated on <https://github.com/dice-group/Palmetto/wiki/How-Palmetto-can-be-used/b1bb4cc5ed63171b85fb2a055c198a087b556837>, version 14. July 2021

³<https://radimrehurek.com/gensim/>

⁴<https://numpy.org/>

⁵<https://www.nltk.org/>

-
- (i) Extract the law paragraphs from the BGB's XML file and divide each paragraph into its sentences, which are then used as initial SCD windows.
 - (ii) Lowercase all characters, tokenize the sentences into words, stem the words, and eliminate stop words from a wordlist containing 232 German words. These four tasks are called preprocessing tasks. Preprocessing a text of a document transforms the text in a more digestible form for machine learning algorithms and increases their performance [VIN15].
 - (iii) Form an initial SCD matrix where each row contains the word probability distribution for one sentence of the corpus.
 - (iv) Apply USEM with one of the three methods greedy, K-Means, or DBSCAN to detect similar rows in the SCD matrix. Afterwards, merge the similar rows or the rows in the same cluster by summing the distributions' values.

We run each method with different hyperparameters influencing the number of SCDs estimated. To be able to show the results of all methods in one figure, we represent the results by the number of SCDs estimated. We show this number of SCDs by the reduction of the number of windows in percent, i.e., if an initial SCD matrix of 8 020 is reduced to 802 rows, the matrix would be reduced by 90 %. For example, in this case K would be 802 for the method K-Means.

- (v) Calculate the UMass coherence using Gensim for the newly estimated SCD matrix on the corpus. Hereby, for each SCD the word probability distribution is used to determine the 20 most probable words of the referenced SCD windows. For each SCD these 20 words are interpreted as the SCD's topic.

For comparison, we train two topic models by LDA using Gensim and the hyperparameters $\alpha = 0.01$ and $\beta = 0.05$. Small α and β lead the model to assign each document a single topic with a high probability, this matches the idea of associating an SCD window with one SCD. We train models with different numbers of topics and represent the number of topics by the reduction of the number of documents given to the model in percent, analogously to the reduction described in (iv) previously.

LDA Windows: This topic model is trained on the 8 020 sentences as documents. Therefore, the model's document topic distributions allow to determine the topic of each sentence and thus the model's topics are comparable to the SCDs referencing multiple sentences in the corpus. However, LDA is not designed to be trained with very short documents like single sentences.

4. USEM – UnSupervised Estimation of SCDs

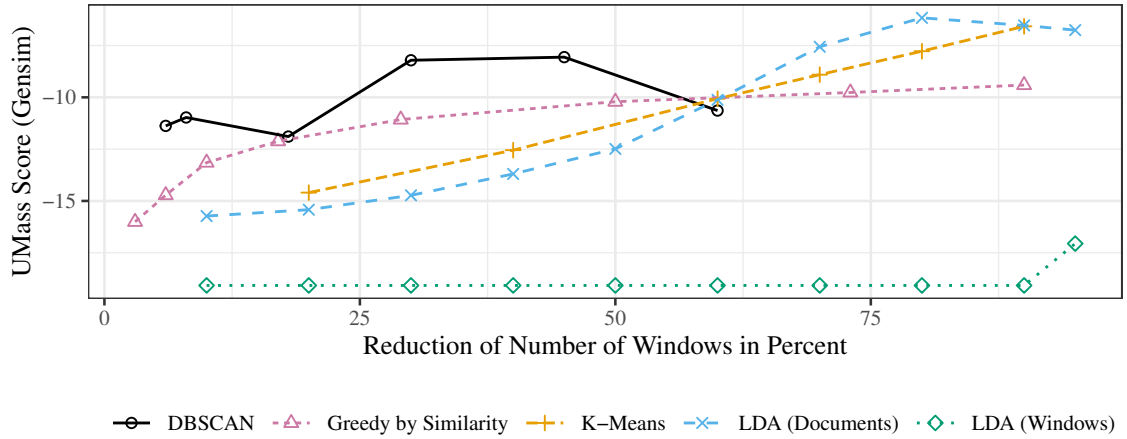


Figure 4.2.: UMass coherence of the three methods using USEM and the coherences of both topic models trained using LDA for comparison. The reduction of the number of windows in percent represents the number of SCDs.

LDA Documents: This topic model is trained on the 2462 law paragraphs as documents and applies LDA in its typical fashion with medium sized documents. However, using the document topic distributions of this model it is not possible to determine the topic of each sentence, as each of the model’s documents contain more than one sentence.

Again, we calculate the UMass coherence for each topic model directly using Gensim’s functionality.

4.3.4. Results

In this section, we present the results gained using USEM and the previously described workflow.

In Figure 4.2, the coherences of the three methods using USEM and both topic models are shown. The UMass scores calculated by Gensim are negative, higher values are better. On the left side, the reduction of the number of windows is small, thus many SCDs are created. Going to the right, the number of SCDs decreases, e.g., the rightmost triangle of greedy similarity represents 834 SCDs gained from initially 8020 windows.

The lines of DBSCAN, greedy similarity, K-Means, and LDA Documents are all close together, while LDA Windows shows poor results far below all other lines. This observation demonstrates that LDA Windows is not capable of estimating SCDs in

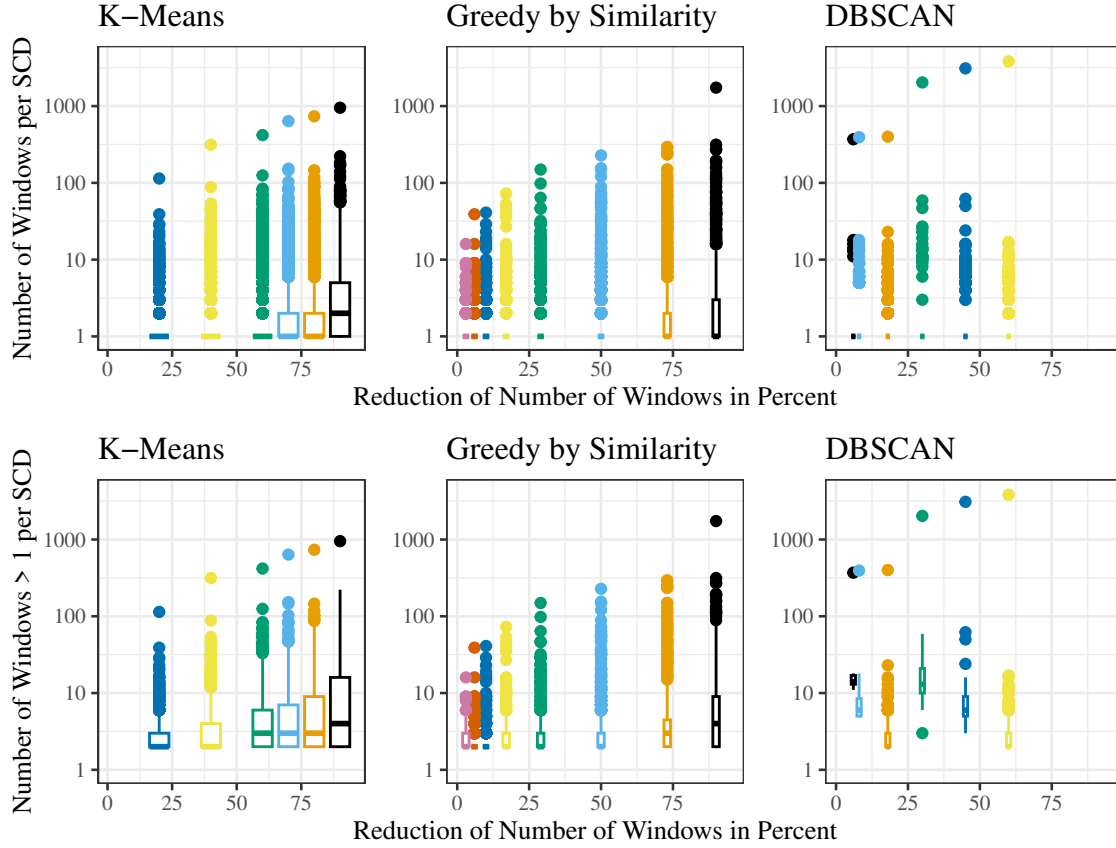


Figure 4.3.: Number of windows referenced by one SCD for the three different methods of USEM. In the lower row, SCDs referencing only one window are omitted.

an unsupervised manner, because the windows used as documents are too small. LDA Documents, however, shows the UMass score a good topic model achieves on the BGB, and USEM using K-Means achieves similarly good values. USEM works well with greedy similarity and less reduction of windows, but K-Means becomes better for more reduction. DBSCAN is quite unstable and the amount of reduction is difficult to configure using the hyperparameters ϵ and number of minimum samples ms . Although, the coherence values of DBSCAN are good, in Figure 4.3 we later see why DBSCAN is not a good choice.

To summarize, using USEM with K-Means yields coherences on par with LDA. However, LDA is not capable of estimating SCDs, which is what USEM is designed for.

In Figure 4.3 for each of the three methods two plots are shown. In the upper row,

for each percentage of reduction the numbers of referenced windows are shown by boxplots on a logarithmic scale. The lower row shows the same, but SCDs referencing only one window are omitted. We focus on the lower row: For K-Means and greedy similarity, most SCDs reference less than 10 windows, which is a good number of references. However, there are also many outliers referencing more windows. For K-Means the largest number of references is 952 and 1 741 with greedy similarity. An SCD referencing 1 741 windows references 21 % of the corpus and it is hard to imagine that 21 % of the corpus share the same concept. Again, these numbers demonstrate that greedy similarity does not work well with a high reduction of the number of windows.

Using DBSCAN there are more SCDs referencing a large number of windows, which also implies that there are many SCDs referencing only one window. Also, the largest number of references is 3 832 for DBSCAN, which means that a single SCD references 48 % of the corpus. An SCD referencing nearly half of the corpus cannot be good, because only this single SCD can reference 48 % of the sentences, while no other SCD can reference the same sentences.

Summarized, K-Means shows an overall very good distribution of referenced windows per SCD and greedy similarity is good, too. DBSCAN should not be used because it generates an SCD that references almost half of the corpus.

4.4. Related Work

Before we conclude the chapter, we take a look at related work. USEM can be understood as a technique for unsupervised corpus annotation. Similarly, OpenIE [AJPM15] extracts *spo*-triples from sentences which may be used as annotation. The evaluation of [BBG⁺21a] and [KBBM21] use OpenIE and Wiktionary⁶ to get an initial set of SCDs. However, both techniques are still supervised to some extent: OpenIE is trained on English language corpora and Wiktionary is composed by humans.

In the context of SCDs, we interpret an SCD as a corpus annotation and in context of this chapter, an SCD annotates multiple sentences of similar concepts all over the corpus' text documents. Topic models assign a distribution over the topics, estimated by the model itself, to each text document in the corpus, and each topic is characterized by a distribution of occurring words in the topic's documents. Thus, similarly to the topics of a topic model, USEM associates text documents with SCDs representing concepts. LDA [BNJ03] is a generative model representing documents as a probability distribution over topics. Many extensions have been proposed to

⁶<https://www.wiktionary.org/>

optimize the performance of LDA, e.g., the author-topic model [RZGSS04], which extends LDA to couple each author of a document with a multinomial over words, and the dynamic topic model [BL06], which allows for analyzing topic changes over time.

Documents assigned with a similar distribution over the topics, are assumed to be similar in terms of an topic model. However, LDA's perception of similar documents may not always match the human perception of similar documents [TRH16].

USEM uses greedy similarity, K-Means, or DBSCAN to identify similar sentences. Another technique to find similar sentences in a corpus of text documents is Similar Short Passages Identifier (SiSP) [SN08]. SiSP first extracts features from the sentences and then creates clusters of similar sentences. The authors evaluate the clusters found by SiSP against human-labeled sentences. USEM may be used with SiSP, however, SiSP was developed for the Portuguese language only.

A further idea is to cluster sentences hierarchically [KR15]. In difference to the clustering techniques used by USEM, the authors start with a sentence in the corpus and build a hierarchical clustering from this sentence. The hierarchical clustering has a tree-like structure, i.e., after starting from the first sentence, the tree branches across multiple levels to different concepts in the corpus.

Clustering can not only be used to identify similar sentences, it can also help to annotate sentences with their sentiment [HSE11]. The authors assume that two sentences in the same cluster have a similar sentiment and thus they can enrich the number of labels in a sparsely labeled corpus. In cases, where short sentences do not contain enough shared words to apply the cosine similarity, a ranking of the suitable clusters for each sentence can be used [YCZS14] to increase the performance of clustering techniques.

Text summarization is another field of research, where clusters of similar sentences are used. Thereby, the idea is to remove most of the similar sentences and only keep one sentence from each cluster. SimFinder [HKH⁺01] clusters small pieces of text, like sentences, into tight clusters. Unlike USEM, SimFinder does not work in an unsupervised way, as it needs feature words in beforehand.

Another approach for text summarization is to extract the word vectors from each sentence and weight each word using techniques in the spirit of tf.idf [Ram03]. Then, the weighted word vectors are clustered using the cosine similarity with K-Means [Llo82], again, from each cluster one sentences makes up the summary [SK17]. This approach overlaps with USEM in using word vectors, the cosine similarity, and K-Means. However, the authors only pursue the goal of text summarization, or in other words, extractive summaries, while USEM uses three methods and represents the concepts and topics of a corpus in the estimated SCD matrix.

4.5. Interim Conclusion

This chapter introduces USEM with three methods, namely K-Means, greedy similarity, and DBSCAN. USEM estimates SCD matrices for corpora of text documents in an unsupervised manner. Thereby, USEM detects sentences of similar concepts or topics in a corpus and then associates the same SCD to these similar sentences. Additionally, a model selection approach is introduced to detect the best method and hyperparameters of USEM for a corpus. Together with the model selection approach, USEM solves Problem I (Subsection 3.2.1) we came across building the SCD-based IR agent.

An SCD matrix for a corpus can be interpreted as a topic model of the corpus. Hence, the well-known LDA is used to evaluate the performance of USEM. We use the UMass coherence to evaluate the quality of each model and show, that especially USEM using K-Means performs as good as LDA. Generally, without a focus on SCDs, USEM provides a new and powerful technique to create a topic model for a corpus. Because DBSCAN associates too many sentences with the same SCD, DBSCAN is not suitable for most use-cases.

In this chapter, we put efforts in finding the referenced sentences of the SCDs. With USEM we get the word probability distribution and the referenced sentences for each SCD. Though, the SCDs estimated by USEM still have an empty set of additional data \mathcal{C} . The next chapter focusses on estimating content for \mathcal{C} , i.e., short descriptions or labels for the SCDs.

5. LESS is More: Label Estimation for SCDs without Supervision

The sections of this chapter are largely taken verbatim from:

- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **LESS is More: LEan Computing for Selective Summaries** in *KI 2023: Advances in Artificial Intelligence. Lecture Notes in Computer Science*, Springer
https://dx.doi.org/10.1007/978-3-031-42608-7_1

Magnus Bender developed the initial idea, conducted the experiments, and wrote the manuscript of the publication. The other three authors fundamentally supervised the research by discussing ideas, proofreading the manuscript multiple times, and giving feedback.

5.1. Introduction

In this chapter, we present a solution to Problem III (Subsection 3.2.4) we came across when building an SCD-based IR agent. USEM estimates initial SCDs for clustering similar sentences across the agent’s—possibly tiny—corpus. However, a cluster is only a set of similar sentences and it is difficult to describe such set to a human user in a comprehensible way. For this purpose, a label or a short description helps the agent to present retrieved clusters and its SCDs more comprehensibly to users. Therefore, each SCD should be associated with a label in addition to the referenced similar sentences. In accordance with the intentions of SCDs, the computation of such labels shall require little computational resources and shall work in an unsupervised way even with tiny corpora.

State-of-the-art LLMs might be applied to compute the labels. However, LLMs like BERT [DCLT19] need specialized hardware to run fast and huge amounts of computational resources resulting in high energy consumption [SGM19]. In addition, BERT uses a pre-trained model that must be trained beforehand on large amounts of data while we are interested in an approach working even with less data, e.g., tiny user supplied corpora.

Existing approaches solve the problem of tiny corpora, e.g., by unifying training data for multiple tasks, like translation, summarization, and classification. Together, the unified data is sufficient to train the model and the model becomes multi-modal for solving the different tasks of the training data. Elnaggar et al. [EGGM18] create such a multi-modal model, and work, similar to our evaluation, with documents about German law, but we are interested in an unsupervised approach requiring no training data at all. In general, a label for an SCD can be estimated by a summarization, i.e., the label is the summarization of a cluster of similar sentences. TED [YZG⁺20] is an unsupervised summarization approach, but it uses a transformer architecture [VSP⁺17] which is the basis for most LLMs. Thus, TED still needs specialized hardware to run fast and huge amounts of computational resources.

Extractive document summarization is a subfield of text summarization, a field in which a summary is created by selecting the key sentences from a document [MC17, ZLWZ18]. However, we are interested in calculating a label for a cluster of similar sentences and not identifying the key sentences of a document. Topics in topic models consist of representative words and each topic represents one topic of the corpus, like one cluster of similar sentences. There exist approaches for computing labels for topics. However, many approaches need supervision [LGNB11, HEGM13, BLB16].

The main problems with the above-mentioned approaches are the need for extensive training data and the need to compute on specialized hardware. Our solution, LESS, is an lightweight approach for Label Estimation for SCDs without Supervision. LESS uses the SCD estimated by USEM and creates labels that can be used to describe the SCDs to humans. We assume that the concept of each SCD is implicitly defined by the content of the sentences referenced, and each label describes the concept of an SCD: So, LESS identifies the best fitting sentence. Together, LESS and USEM associate any corpus with labelled SCDs, where each SCD references similar sentences of the same concept, has a label describing its concept, and an SCD-word distribution. LESS in conjunction with USEM neither needs specialized hardware nor additional training data. In the evaluation, LESS computes labels with less time and computational resources while providing similar results as BERT.

The remainder of this chapter is structured as follows: First, we formalize the problem of computing labels for SCDs and provide our solution LESS. Second, we evaluate the performance of LESS against the well-known BERT and demonstrate that LESS is on par with an approach using BERT, while being lean and requiring less resources and no pre-trained models. Finally, we conclude with an interim summary.

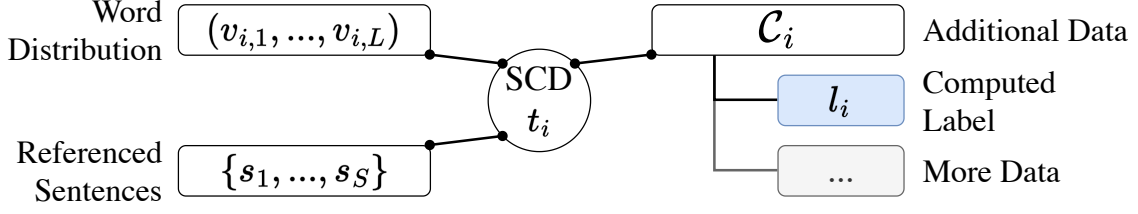


Figure 5.1.: An SCD t_i with its intra-SCD relations to various parts forming the SCD itself. LESS uses the word distribution and the referenced sentences to compute labels.

5.2. Computing Labels for SCDs

For a better understanding of the proposed approach, we first take a look at the different relations among SCDs. Afterwards, we introduce how LESS solves Problem III of computing an SCD's label.

5.2.1. Relations and SCDs

There are two different types of relations among SCDs. First, there are relations between SCDs, e.g., to model complementarity between two SCDs (considered in Chapter 8). Second, and faced in this chapter, are the relations within an SCD to its various parts, which together form the SCD. The SCD t_i has the SCD-word distribution $(v_{i,1}, \dots, v_{i,L})$, the matrix' row, and the referenced sentences $\{s_1, \dots, s_S\}$. In this chapter, we are interested in the label l_i . The label l_i is part of SCD's additional data \mathcal{C}_i , in this chapter the only element in \mathcal{C}_i . The various parts which form the SCDs together can be seen in Figure 5.1.

Therefore, our setting is that there are intra-SCD relations to various parts and inter-SCD relations to other SCDs. Additionally, relations can be added by storing data or references in \mathcal{C}_i , e.g., inter-SCD relations may be added as references to other SCDs (considered in Chapter 8).

5.2.2. Labels to Select From

Given that LESS builds upon USEM, the SCD t_i , consisting of the SCD-word distribution $(v_{i,1}, \dots, v_{i,L})$ and the referenced sentences $\{s_1, \dots, s_S\}$, is the input for which LESS computes a label. Thus, the label of the SCD needs to be computed only based on these two parts or additional supervision would be needed. In general, a

good label could be a short summary given the word distribution of the SCD, based on the assumption that the word distribution generates the sentences. Therefore, we look for a short sentence, i.e., without many filler words, that is close to the word distribution. We define a *utility* function in the next Subsection 5.2.3 to measure how well a candidate for a label fits the concept described by an SCD.

The problem to be solved can be formulated as follows:

$$l_i = \arg \max_{l_j \in \text{all possible labels}} \text{Utility}(l_j, ((v_{i,1}, \dots, v_{i,L}), \{s_1, \dots, s_S\}) \text{ of } t_i)$$

The computed label l_i for t_i (currently consisting of the word distribution and the referenced sentences) is the label with the highest utility. Now, there are two points to address (i) is it not possible to iterate over *all possible labels* and (ii) what is a label with a high utility.

For the first point, we have to specify how a label should look like. A label is a sequence of words like a short description. We argue that a sentence straight to the point, i.e., without many filler words, is a good description and thus a good candidate for a label. Furthermore, each SCD has a set of referenced sentences which together represent the concept which is represented by the SCD. Thus, we use the referenced sentences $\{s_1, \dots, s_S\}$ as set of possible labels for each SCD.

Using sentences from the corpus and not generating sentences, e.g., with an LLM like GPT [BMR⁺20], has multiple benefits: No pre-trained model or training data is needed while the sentences used as labels will still match the style of writing in the corpus. Additionally, no computational resources for GPT are needed and the sentences must not be checked for erroneous or other troublesome content, e.g., LLMs may degenerate into toxic results even by seemingly innocuous inputs [GGS⁺20].

The problem can now be reformulated as:

$$l_i = \arg \max_{s_j \in \{s_1, \dots, s_S\}} \text{Utility}(s_j, (v_{i,1}, \dots, v_{i,L}))$$

The computed label l_i for SCD t_i is the referenced sentence of the SCD which provides the highest utility. The utility function now only gets the word distribution as input because the referenced sentences are already used as set of possible labels. Thus, LESS computes for each sentence its utility given the word distribution and takes the best. Again, an LLM like BERT may be used to calculate the utility. However, we are interested in a lean computing approach.

5.2.3. Utility of Sentences as Labels

The utility describes, by a value between 0 and 1, how well a sentence fits the concept described by the SCD. The referenced sentence with the highest utility is assumed to be a good label for the SCD in human interception.

As mentioned before, we assume that each SCD's word distribution generates the referenced sentences, then the best label for this SCD is a sentence that is most similar to the word distribution, in terms of the concept represented. Thus, the cosine similarity allows to determine the similarity between two vectors and a word distribution can be interpreted as a word vector. Thus, we define the utility function as the cosine similarity between the SCD-word distribution and the referenced sentence's word vector. In addition, the cosine similarity has proven to be a good choice for identifying sentences with similar concepts by using their word distributions: The MPS²CD algorithm uses the cosine similarity to identify the best SCD for a previously unseen sentence based on its word vector. Thus, we use the most similar referenced sentence by cosine similarity as the computed label of an SCD.

Altogether, the lean computation of a label for an SCD can be formulated as:

$$l_i = \arg \max_{s_j \in \{s_1, \dots, s_S\}} \frac{\vec{s}_j \cdot v_i}{\|\vec{s}_j\|_2 \cdot \|v_i\|_2}$$

The word vector of each referenced sentence s_j is represented by \vec{s}_j and the SCD-word distribution by v_i . In general, the cosine similarity yields results between -1 and 1 , in this case all inputs are positive and thus the utilities are between 0 and 1 only.

Finally, the computed sentence to become the label may be slightly post-processed, s.t., it becomes a sentence straight to the point without many filler words. This can be achieved by removing stop words.

5.2.4. Algorithm LESS

Based on the two previous subsections, LESS is formulated in Algorithm 5. LESS first estimates the SCD matrix using USEM, a step that might be skipped if an SCD matrix is supplied, and initializes an empty SCD set $g(\mathcal{D})$. In Lines 6 - 11, the label l_i is computed for each of the K SCDs t_i iterating over the rows of the SCD matrix. First, the SCD-word distribution v_i is extracted and also all candidates for the label—the referenced sentences of each SCD—are fetched from the corpus. In Line 9 the label is computed as described in Subsection 5.2.3. Finally, the associated SCD t_i is composed, containing additional data in \mathcal{C}_i , namely a computed label l_i and the referenced sentences, and t_i is added to the SCD set $g(\mathcal{D})$.

Algorithm 5 LEan computing for Selective Summaries

```

1: function LESS( $\mathcal{D}$ )
2:   Input: Corpus  $\mathcal{D}$ 
3:   Output: SCD matrix  $\delta(\mathcal{D})$ ; SCD set  $g(\mathcal{D})$  containing labels  $l_i$  for SCDs  $t_i$ 
4:    $\delta(\mathcal{D}) \leftarrow \text{USEM}(\mathcal{D})$  ▷ Run USEM
5:    $g(\mathcal{D}) \leftarrow \{\}$  ▷ Initialize empty SCD set  $g(\mathcal{D})$ 
6:   for each row of matrix  $i = 1, \dots, K$  do
7:      $v_i \leftarrow g(\mathcal{D})[i]$  ▷ Extract SCD-word distribution
8:      $\{s_1, \dots, s_S\} \leftarrow \text{REFERENCEDSENTENCES}(i)$  ▷ Get referenced sentences
9:      $l_i \leftarrow \arg \max_{s_j \in \{s_1, \dots, s_S\}} \frac{\vec{s}_j \cdot v_i}{\|\vec{s}_j\|_2 \cdot \|v_i\|_2}$  ▷ Compute label
10:     $t_i \leftarrow (\{l_i\}, \{s_1, \dots, s_S\})$  ▷ Compose associated SCD with computed label
11:     $g(\mathcal{D}) \cup \{t_i\}$  ▷ Add to SCD set
12:   return  $\delta(\mathcal{D}), g(\mathcal{D})$ 

```

Next, we present an evaluation of LESS and compare the results to an approach using BERT for computing labels for SCDs.

5.3. Evaluation

After we have introduced LESS, we present an evaluation. First, we describe the used corpus, two approaches using BERT to compute labels, and the evaluation metrics. Afterwards, we present the results of the evaluation and show the performance and runtime of LESS in comparison to BERT.

5.3.1. Dataset

In this evaluation we use the Bürgerliches Gesetzbuch (BGB)¹, the Civil Code of Germany, in German language as corpus. The BGB does not provide enough training data to train a specialized BERT model. Additionally, BGB does not provide labels which can be used for supervised training.

Given the vast amount of text written in English and the fact that English is the language of computer science, most natural language processing techniques work better with the English language. The BGB is therefore a *difficult* dataset and a good example to use with approaches such as LESS that require only little data and

¹<https://www.gesetze-im-internet.de/bgb/>, English translation https://www.gesetze-im-internet.de/englisch_bgb/

no supervision. To apply BERT on the BGB, we have to rely on external pre-trained models that have been trained on other data.

As said before, the BGB is freely available and can be downloaded as XML file. Therefore, it is easily parsable and processable. As the corpus is a law text it consists of correct language, i.e., punctuation and spelling follow the orthographic rules. Thus, little preprocessing and no data cleaning is needed.

The entire corpus consists of 2 466 law paragraphs and overall 11 904 sentences which are used as SCD windows. Each law paragraph contains between 1 and 49 sentences with an average of 4.83 sentences. The vocabulary consists of 5 315 words, where each sentence is between 1 and 35 words long with an average of 7.36 words. These numbers are different to Subsection 4.3.1 because we changed the preprocessing and partitioning of sentences and updated the version of the BGB.

5.3.2. Approaches using BERT

We evaluate LESS against two approaches using BERT to compute labels for SCDs. Thereby, BERT is compared to LESS in terms of runtime and actual content of the labels.

BERT is used as a different utility function to select the best sentence as label from the set of referenced sentence for each SCD. We do not use freely generated texts, e.g., by GPT, as labels because these labels need to be checked for erroneous content as already stated in Subsection 5.2.2. Additionally, comparing freely generated text to a label selected from a set of referenced sentences is like comparing apples and oranges.

The two approaches using BERT work as follows:

BERT Vectors works similar to LESS, but uses the embeddings produced by BERT instead of the word distributions of each sentence. Throughout all referenced sentences an average embedding of each SCD is calculated. Then the referenced sentence with the most similar embedding to the average embedding in terms of cosine similarity is used as label. Thereby, the embedding of the CLS token, representing the entire sentence instead of a single token, for each sentence from the pre-trained model `bert-base-german-cased`² is used.

BERT Q&A uses the ability of BERT to answer a question. Thereby, BERT gets a question and a short text containing the answer. The assumed answer is then highlighted by BERT in the short text. We use the fine-tuned model

²<https://huggingface.co/bert-base-german-cased>

`bert-multi-english-german-squad2`³ and compose our answer and short text by concatenating all referenced sentences of each SCD. Hence, BERT highlights a referenced sentence or a part of one while the question consisting of all referenced sentences asks BERT to represent all sentences.

As we do not have supervision for our corpus, we cannot fine-tune BERT models for label computation.

5.3.3. Hardware and Metrics

LESS and both approaches using BERT run in a Docker container. The evaluation with off-the-shelf hardware is done on a machine featuring 8 Intel 6248 cores at 2.50GHz (up to 3.90GHz) and 16GB RAM, referred to as CPU. However, this virtual machine does not provide a graphics card for fast usage of BERT. Thus, all experiments using BERT are run as well on a single NVIDIA A100 40GB graphics card of an NVIDIA DGX A100 320GB, referred to as GPU. Beneath, the NVIDIA Container Toolkit is used to run our Docker container with NVIDIA CUDA support.

The runtime of the approaches is measured in seconds needed to compute all labels for the BGB. Thereby, the initialization of the BERT models is excluded but the necessary transformations of the referenced sentences are included. These transformations include the tokenization for BERT and composition of word distributions for LESS. SCDs referencing only a single sentence do not require a computation, and the single sentence is used as label.

The performance is measured by the agreement between LESS and the results of each of the two BERT-based approaches. Specifically, for one BERT-based approach, all SCDs where BERT and LESS compute the same label are counted and divided by the total number of SCDs. We distinguish between considering all SCDs, including those referencing only one sentence, or excluding those SCDs with only one referenced sentence.

5.3.4. Workflow and Implementation

LESS and the BERT-based approaches are implemented using Python. The implementations use the libraries Gensim⁴, NumPy⁵, and Huggingface Transformers⁶.

³<https://huggingface.co/deutsche-telekom/bert-multi-english-german-squad2>

⁴<https://radimrehurek.com/gensim/>

⁵<https://numpy.org/>

⁶<https://huggingface.co/docs/transformers/>

LESS is optimized to run on a single core and does not offer multi-core capabilities. BERT uses all available cores or a graphics card.

We evaluate LESS on ten similar SCD matrices. The SCD matrices are estimated by USEM with methods greedy similarity and K-Means. Each method is run with five different hyperparameters: 0.8, 0.7, 0.6, 0.5, 0.4 for greedy similarity and 0.8, 0.6, 0.4, 0.3, 0.2 for K-Means. The evaluation workflow for each of the ten matrices follows:

- (i) Run USEM and thus estimate the SCD matrix and all SCDs, lacking labels. This step includes fetching the BGB’s XML file and running preprocessing tasks [VIN15]. Depending on the hyperparameters USEM estimates between 2 159 and 10 415 SCDs for the corpus.
- (ii) Run LESS to compute labels for the SCDs. Here, the runtime of LESS is captured.
- (iii) Run BERT Q&A and BERT Vectors to compute two more sets of labels for the SCDs. Again, the runtime is captured, separately for each approach and for CPU and GPU.
- (iv) Calculate the agreement between LESS and BERT Q&A and BERT Vectors as described in Subsection 5.3.3. Thereby, use the set of labels computed by each BERT-based approach and LESS. There is practically no difference between the labels computed by BERT on the GPU and on the CPU. Thus, we do not differentiate between CPU and GPU in terms of agreement.

5.3.5. Results

In this section, we present the results gained using LESS in comparison to the BERT-based approaches and the previously described workflow.

In the left part of Figure 5.2, the runtimes of LESS, BERT Q&A, and BERT Vectors are displayed. The values are averaged over all ten evaluated SCD matrices and are shown on a square root scale. Comparing both BERT-based methods, BERT Vectors is always faster than BERT Q&A, especially when running on CPU. LESS is on off-the-shelf hardware as fast as BERT Vectors on the A 100 GPU and always faster than BERT Q&A. The different computational resources needed by BERT and LESS are good to grasp by comparing the runtimes of on the CPU. LESS utilizes only one core while BERT uses eight cores and still BERT is significantly slower. Thus, LESS needs less time on less cores to compute the labels. Typically distilled LLMs represent lean computing, but distilled LLMs still need to run on a GPU to be fast, while LESS remains lean.

5. LESS is More – Label Estimation for SCDs without Supervision

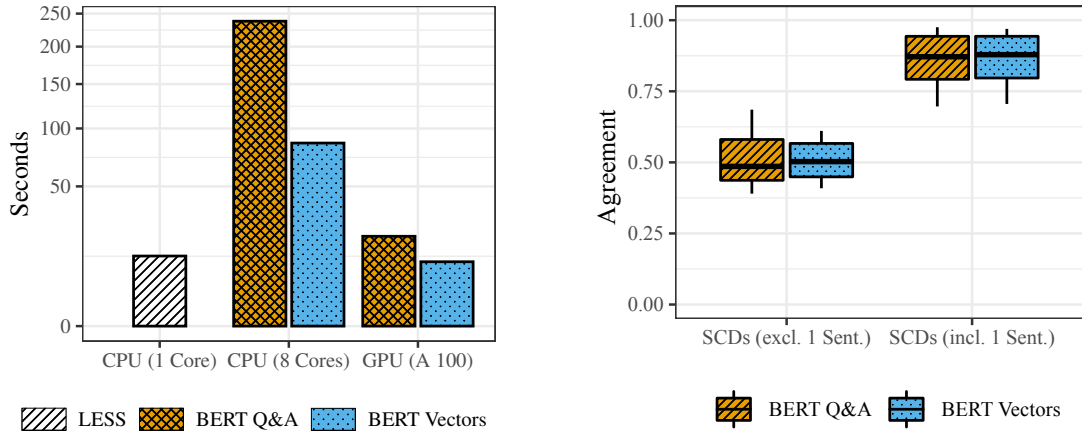


Figure 5.2.: Left: Runtime of LESS, BERT Q&A, and BERT Vectors with a square root scale. Right: Agreement of BERT Q&A and BERT Vectors (compared to LESS respectively) divided by considering all SCDs or excluding those with one referenced sentence.

In the right part of Figure 5.2, the agreements over all ten evaluated SCD matrices are shown with boxplots. We divide between considering all SCDs or only SCDs with more than one referenced sentence. There are no big differences between BERT Q&A and BERT Vectors. Considering all SCDs gives very good agreements, which is particularly important as we are interested in labels for all SCDs. At first glance, the agreements considering only SCDs with more than one referenced sentence do not look good but we have to look at the number of the referenced sentences of each SCD.

In order to better rate the agreement values, we calculate the agreement for random sentence, which is the theoretical agreement a random approach would result in. This random approach randomly chooses for each SCD which referenced sentence becomes the label, i.e., the agreement for random sentence is 0.1 for an SCD having 10 referenced sentences. In Figure 5.3, the agreement for random sentence is added as baseline to rate the agreements of BERT and LESS. Besides random sentence, we have the agreements for SCDs including single sentences and excluding single sentences already known from Figure 5.2. The three agreements are displayed for each of the ten SCD matrices and we do not differentiate between BERT Q&A and BERT Vectors as the values do not show a relevant difference. Thus, LESS is able to compute labels for all SCD matrices estimated by USEM and does not depend on specific hyperparameters.

For the leftmost bar in Figure 5.3, randomly selecting a sentence from the clustered sentences results in an agreement of around 0.35. By excluding SCDs with single

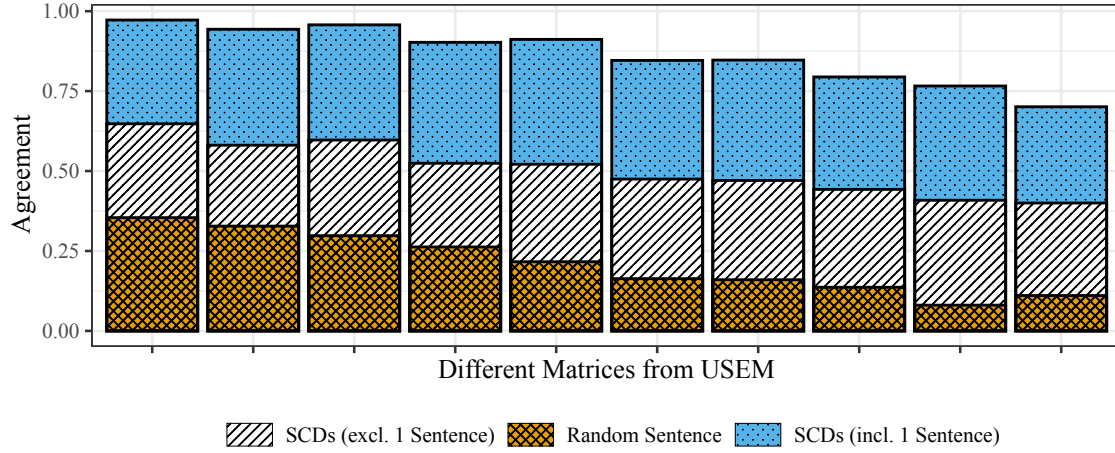


Figure 5.3.: Theoretical average agreement of an approach that randomly selects a sentence as label compared to the agreement of LESS and BERT.

sentences, LESS and BERT agree 68 % of the time and by including all SCDs nearly all the time (97 %).

Implicitly, the agreement for random sentence shows the number of referenced sentences the SCDs have. The SCD matrices on the left side of Figure 5.3 have fewer referenced sentences per SCD as the ones on the right side. With an increasing number of sentences, there are more sentences to select from and the computation becomes more difficult, i.e., it is *easier* to correctly select an item from a set of three than of ten items. This increasing difficulty to the right is also demonstrated by the fact that all agreements become smaller to the right.

The agreement for random sentence is always clearly the lowest, with both other agreements following at some distance. Thus, BERT and LESS achieve a high level of agreement. In summary, LESS computes good labels requiring much less computing resources and computing time.

5.4. Interim Conclusion

This chapter presents LESS and LESS is more. LESS is an unsupervised lean computing approach to compute labels for SCDs. LESS works on any corpus and does not require training data. LESS only needs clusters of similar sentences, which are contained in SCDs and are estimated in an unsupervised way by USEM. Hence, together with USEM, LESS can generate SCDs with labels for any corpus to help

IR agents. We evaluate LESS against two approaches using an LLM, in this case BERT. The evaluation shows that LESS requires significantly less computational resources. Furthermore, LESS does not need any training data. Therefore, we evaluate LESS in a setting, where no training data is available. Hence, we cannot fine-tune a BERT model for our needs and evaluate LESS against two approaches using already fine-tuned BERT models. The labels computed by the BERT-based methods significantly coincide with those of LESS. Summarized, LESS computes good labels needing less computational resources.

The techniques from the last two chapters give the SCD-based IR agent a possibility to process user supplied corpora. The agent can compute SCDs with labels for any corpus without needing additional data. The next two chapters focus on optimizing the computed SCDs and the labels by updating the matrix incrementally and efficiently based on feedback of users.

6. FrESH: Feedback-reliant Enhancement of SCDs

The sections of this chapter are largely taken verbatim from:

- Magnus Bender, Kira Schwandt, Ralf Möller, Marcel Gehrke: **FrESH – Feedback-reliant Enhancement of Subjective Content Descriptions by Humans** in *Proceedings of the Humanities-Centred AI (CHAI) Workshop at KI2023, 46th German Conference on Artificial Intelligence, 2023*
<https://ceur-ws.org/Vol-3580/paper3.pdf> (Slides: <https://dx.doi.org/10.25592/uhhfdm.13423>)

The publication itself is based on Kira Schwandt’s bachelor thesis, which is written in German. Magnus Bender and Ralf Möller fundamentally supervised the bachelor thesis by discussing ideas, proofreading the manuscript multiple times, and giving feedback. For the publication: Magnus Bender developed the initial idea and wrote the manuscript in English based on the results of the bachelor thesis. Marcel Gehrke supervised the research by proofreading the manuscript.

6.1. Introduction

Typically, in machine learning, a model is trained to perform specific tasks on the model rather than directly on the data. Training a model is an expensive task. However, if data points later on have to be removed, our goal is to discreetly manipulate the model instead of retraining it from scratch. For example, it may turn out that an item from the data is erroneous or there may be privacy or copyright related problems with an item. Additionally, the model may produce wrong results because some item may be associated incorrectly. In all these cases, the item needs to be removed from the data and the model. However, removing an item from a model is a difficult task because the model only encodes the data and does not contain the data in such a way that individual elements can be distinguished and removed. Therefore, models are often retrained from scratch after items are removed from the training data.

For different models, there are different approaches to avoid retraining from scratch, e.g., for K-Means [Llo82, GGVZ19] or linear and logistic regression [ISCZ21]. The

common idea is to avoid retraining the model and instead updating the model by applying an inverse operation that removes an item of the data from the model.

In this dissertation, we build an SCD-based IR agent. We identified Problem II (Subsection 3.2.3) as the difficulty of updating the used SCDs on the basis of feedback: From the perspective of a human user of the agent, there may be a faulty item in the IR agents response. In this situation, the human user may give feedback to the agent and the agent needs to update its SCD matrix by removing the faulty association between sentence and SCD. Hence, the SCDs should be updated and the IR agent should not need to retrain the SCD matrix from scratch after the sentence has been removed from the corpus.

To solve Problem II, this chapter provides the first part, FrESH, an approach for Feedback-reliant Enhancement of Subjective Content Descriptions. The IR agent may use FrESH to process feedback by removing a faulty sentence from the SCD matrix and only needs to update the SCD the faulty sentence has been associated with. In particular, FrESH can be used to remove sentences from the SCD matrix if their content should be removed entirely, i.e., to keep *Fake-News* out of the corpus or to remove copyright or privacy protected content.

Adding a feedback mechanism to the IR agent allows the users to improve their own subjective model by enhancing an initial SCD matrix learned by USEM and LESS on their corpus. For example, FrESH makes it easier for a scientist in the field of humanities to build a custom SCD based model for their corpora. A corpus can be automatically annotated with SCDs, and then it is optimized and corrected using FrESH by feedback from the scientist. Without FrESH, it would not be possible to optimize and correct a corpus automatically annotated with SCDs, thus requiring more manual work by the scientist. Additionally, in the humanities corpora of text documents are often rather small. Therefore, techniques like LLMs cannot be applied in a considerable number of cases.

The remainder of this chapter is structured as follows: First, we formalize the problem of incorporating feedback by removing false associations of sentences with SCDs in an SCD matrix. Afterwards, we provide three consecutive methods to solve the problem and evaluate each. Finally, we conclude with an interim summary.

6.2. Incorporate Feedback

In this section, we present FrESH and thereby how to incorporate human feedback to enhance an SCD matrix. For our method, we assume that the feedback exactly states which sentence is falsely associated with its SCD, i.e., the human user may click on a button *remove sentence*. Therefore, we consider the problem of removing

		SCD matrix	
		Counts	Distribution
Errors/ Feedback	Faulty sentences and their SCDs	Method 1	Method 3
	Only faulty sentences	Method 2	Method 4

Figure 6.1.: The four different cases regarding the input data of FrESH.

a faulty sentence from an SCD matrix. This also removes the sentence from the corpus, making FrESH useful for removing *Fake-News* and copyright or privacy protected content, too.

In this chapter, we differentiate between two types of SCD matrices: The values contain the counts of the words without normalization $\delta(\mathcal{D})$ or a normalized version containing row-wise distributions of words $\delta_{||\cdot||}(\mathcal{D})$. Additionally, we differentiate between the way the errors are contained in the feedback: Only a faulty sentence is given as input or a faulty sentence together with its SCD is given. In both cases a set p contains either sentences s or pairs of sentences s and SCDs t .

In total, we have four different cases regarding the input data of FrESH, which are also shown in Figure 6.1. Next, we will consider each case and develop method 1 to 4 to solve each.

6.2.1. Method 1

The input consists of an SCD matrix containing the counts of the words and a set p of faulty sentences with their associated SCDs. It is then possible to revert the operations of SEM (Algorithm 1). SEM adds in Line 9 for each word in a sentence the count of the word weighted by an influence value to the row of the matrix representing the SCD. The first method (M1) reverses this addition by subtracting the same value in Line 6 of Algorithm 6.

M1 inverts the operations of SEM. Therefore, an SCD matrix learned by SEM on the corpus without p will be identical to an SCD matrix from which p was removed by M1.

6.2.2. Method 2

The input consists of an SCD matrix containing the counts of the words and a set p of faulty sentences. Hence, first the SCD for each faulty sentence needs to be found,

Algorithm 6 FrESH Method 1 (M1)

```
1: function FRESH-M1( $\delta(\mathcal{D})$ ,  $p$ )
2:   Input: SCD matrix  $\delta(\mathcal{D})$ ; Set of faulty sentences with SCDs  $p$ 
3:   Output: Updated SCD matrix  $\delta(\mathcal{D})$ 
4:   for each  $(s, t) \in p$  do                                 $\triangleright$  Iterate over faulty sentences with SCDs
5:     for each  $w_i \in s$  do                                     $\triangleright$  Iterate over words
6:        $\delta(\mathcal{D})[t][w_i] -= I(w_i, s)$                      $\triangleright$  Assume uniform, i.e.,  $I(w_i, s) = 1$ 
7:   return  $\delta(\mathcal{D})$ 
```

Algorithm 7 FrESH Method 2 (M2)

```
1: function FRESH-M2( $\delta(\mathcal{D})$ ,  $p$ )
2:   Input: SCD matrix  $\delta(\mathcal{D})$ ; Set of faulty sentences without SCDs  $p$ 
3:   Output: Updated SCD matrix  $\delta(\mathcal{D})$ 
4:   for each  $s \in p$  do                                        $\triangleright$  Each faulty sentence
5:      $t = \text{MPS}^2\text{CD}(\delta(\mathcal{D}), s)$                          $\triangleright$  Use MPS2CD to get SCD of sentence
6:     for each word  $w_i \in s$  do                                $\triangleright$  Iterate over words
7:        $\delta(\mathcal{D})[t][w_i] -= I(w_i, s)$ 
8:   return  $\delta(\mathcal{D})$ 
```

afterwards the sentence can be removed from the matrix as in M1. The MPS²CD algorithm is used to find a most suitable SCD for a sentence from the available SCDs in Algorithm 7 (M2).

Unlike M1, M2 does not guarantee to produce identical matrices. If a matrix is first trained on a corpus with faulty sentences that are then removed by M3, the resulting matrix may be different from a matrix trained directly on a corpus without faulty sentences. This is caused by the fact, that MPS²CD is used to find a most suitable SCD, which may not be the SCD used initially. However, MPS²CD works quite accurately, so we expect only a small difference, which should not affect an agent using SCDs.

6.2.3. Method 3

The input consists of an SCD matrix $\delta_{\parallel\cdot\parallel}(\mathcal{D})$ containing row-wise distributions of words and a set p of faulty sentences with their associated SCDs. The difficulty is to revert the operations of SEM on the distributions of each SCD. During normalization of the SCD matrix, all counts in each matrix's row are divided by a divisor. To approximate this divisor with M3, we assume that in each row at least one value had the count one, i.e., there was a word that occurred only once in the referenced sentences of the SCD. This word will have the minimal value in the distribution and

Algorithm 8 FrESH Method 3 (M3)

```

1: function FrESH-M3( $\delta_{\|\cdot\|}(\mathcal{D})$ ,  $p$ )
2:   Input: Normalized SCD matrix  $\delta_{\|\cdot\|}(\mathcal{D})$ ; Set of faulty sent. with SCDs  $p$ 
3:   Output: Updated normalized SCD matrix  $\delta_{\|\cdot\|}(\mathcal{D})$ 
4:   for each  $(s, t) \in p$  do ▷ Iterate over faulty sentences with SCDs
5:      $m = \min_{j=1, \dots, L; \delta_{\|\cdot\|}(\mathcal{D})[t][j] > 0} \delta_{\|\cdot\|}(\mathcal{D})[t][j]$  ▷ Minimal value in SCD's row
6:     for each word  $w_i \in s$  do ▷ Iterate over words
7:        $\delta_{\|\cdot\|}(\mathcal{D})[t][w_i] -= I(w_i, s) \cdot m$ 
8:   return NORMALIZEROWS( $\delta_{\|\cdot\|}(\mathcal{D})$ )

```

we assume it had a count of one. So the normalized value is equal to one divided by the divisor, which in turn is a factor that we can use to decrease the values that are subtracted from the matrix by the same amount as during the normalization.

Again, M3 is based on an approximation and may not produce identical matrices compared to those generated with learning from scratch.

6.2.4. Method 4

The input consists of an SCD matrix containing row-wise distributions of words and a set of faulty sentences. For this input, the ideas from M2 and M3 can be combined. First, the SCD for each faulty sentences needs to be determined, which can be done via MPS²CD. Second, the minimal value of this SCD's distribution can be determined and used as factor while changing the SCD matrix.

We will not consider M4 in detail, as it is only a combination of M2 and M3. In most use-cases there is no benefit in normalizing the SCD matrix, because mostly the cosine similarity is used and it does a normalization by definition. Therefore, especially M2 is beneficial for our agent using SCDs and getting feedback from its users, e.g., to remove faulty sentences.

Next, we present an evaluation of M1, M2, and M3.

6.3. Evaluation

After we have introduced FrESH to remove sentences from an SCD matrix, we present an evaluation of the methods M1, M2, and M3. First, we describe the used corpus and evaluation workflow. Afterwards, we present the results of the evaluation.

6.3.1. Dataset

In this evaluation we use the 20 newsgroups¹ dataset. 20 newsgroups is a well-known corpus consisting of e-mails from 20 e-mail newsgroups. Thematically, the 20 newsgroups can be divided into six topics, *computer*, *sport*, *science*, *politics*, *religion* and *for sale*. The entire corpus consists of 18 828 text documents. The documents have between 1 and 39 682 words with a median of 160 words.

6.3.2. Workflow and Metrics

FrESH is implemented in Python and runs in a Docker container on a machine featuring 8 Intel 6248 cores at 2.50GHz (up to 3.90GHz) and 16GB RAM.

To evaluate each method, we first create the full corpus \mathcal{D}_f containing all documents from 20 newsgroups. Each sentence gets annotated by OpenIE [AJPM15] and the extracted *spo*-triples are used as data for the SCDs. Then, we split \mathcal{D}_f into two subsets, \mathcal{D}_s contains the sentences to subtract—considered faulty—and \mathcal{D}_k contains the non-faulty sentences to keep. For each corpus we learn an SCD matrix using SEM, i.e., $\delta(\mathcal{D}_f)$ and $\delta(\mathcal{D}_k)$. Afterwards, we apply one of the three methods to $\delta(\mathcal{D}_f)$ and remove the sentences in \mathcal{D}_s . This step yields $\delta'(\mathcal{D}_f)$, which should be identical to $\delta(\mathcal{D}_k)$.

As a metric, we then calculate the difference of the matrices $\delta'(\mathcal{D}_f)$ and $\delta(\mathcal{D}_k)$ for each SCD t using the Hellinger distance [Hel09]. For calculating the Hellinger distance, the matrices are normalized row-wise to become a distribution.

$$\text{HD}_t(\delta'(\mathcal{D}_f), \delta(\mathcal{D}_k)) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^L \left(\sqrt{\delta'(\mathcal{D}_f)[t][i]} - \sqrt{\delta(\mathcal{D}_k)[t][i]} \right)^2}$$

6.3.3. Results

In this section, we present the results of the evaluation. In Figure 6.3, the duration of removing one sentence with the three different methods is shown. M1 is very fast, while M2 and M3 need more time. M2 needs to determine the SCD via MPS²CD which takes time and M3 needs to calculate the factor needed to maintain the distribution in the matrix. However, all methods are reasonably fast, as it takes at most 4.5ms to remove a sentence which will be clearly faster than retraining the matrix from scratch.

¹<http://qwone.com/~jason/20Newsgroups/>

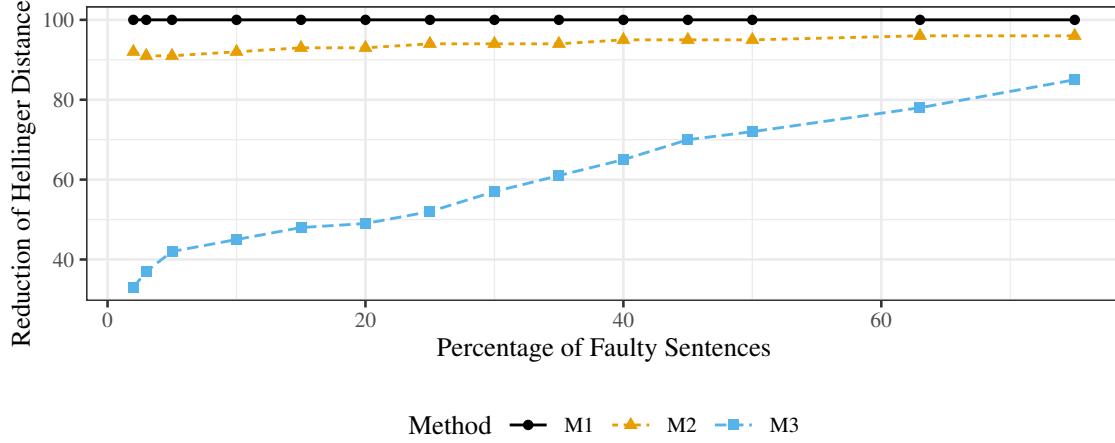


Figure 6.2.: Reduction of difference between the matrices $\delta'(\mathcal{D}_f)$ and $\delta(\mathcal{D}_k)$ for the three methods and different numbers of faulty sentences.

In Figure 6.2, the difference between the matrices $\delta'(\mathcal{D}_f)$ and $\delta(\mathcal{D}_k)$ is shown. The difference is shown as percentage of the reduction of the Hellinger distance averaged for all SCDs: First, the Hellinger distance between $\delta(\mathcal{D}_f)$ (full corpus) and $\delta(\mathcal{D}_k)$ (non-faulty part) is calculated. When removing the faulty sentences from $\delta(\mathcal{D}_f)$ to get $\delta'(\mathcal{D}_f)$ the distance should become smaller, which is shown as reduction. A reduction of 100% implies that $\delta(\mathcal{D}_k)$ equals $\delta'(\mathcal{D}_f)$. The reductions are shown for all three methods and different numbers of faulty sentences in \mathcal{D}_s . The x-axis represents the size of \mathcal{D}_s as percentage of the entire corpus \mathcal{D}_f .

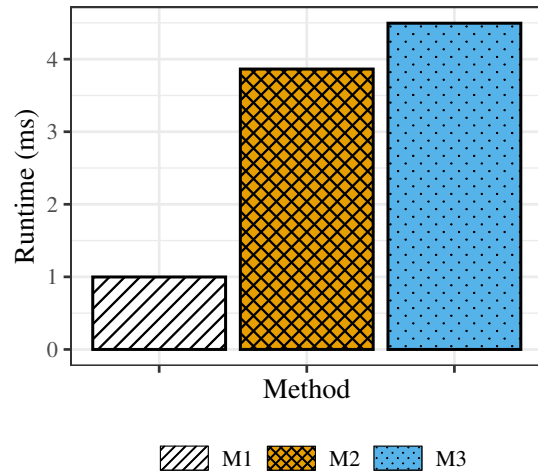


Figure 6.3.: Average runtime removing one sentence with the different methods.

M1 inverts the operations of SEM, therefore, after removing all faulty sentences, both SCD matrices are equal and the difference is reduced completely. M2 reaches very high reductions around 95% and thus provides a reliable technique to remove faulty sentences. The results of M3 are found to be significantly below M1 and M2. Especially, if only few faulty sentences are deleted, the difference keeps to be high. This high difference implies that our assumption made in M3, to approximate the factor used to normalize the matrix,

does not hold.

Therefore an agent maintaining an SCD matrix should store it using counts instead of distributions. Doing so, the agent may use M1 and M2 to incorporate feedback. In particular, storing counts is advantageous because FrESH is more accurate, counts can be stored as integers instead of possibly inaccurate floats, and for most use-cases of the SCD matrix, normalization is done by definition, e.g., as part of the cosine similarity.

Summarizing, an IR agent using SCDs can automatically train an SCD matrix storing counts using USEM on a corpus. This SCD matrix is automatically trained and may contain some faulty sentences and SCDs. When a user uses the IR agent, a button to remove faulty sentences is available, which then triggers FrESH to remove a sentence. Depending on the specific situation, the SCD of a sentence to be removed may or may not be known, and M1 or M2 may be used. Thus, the SCD matrix is updated incrementally and optimized for the user. A limitation of FrESH is that it removes a sentence completely from the SCD matrix, whereas sometimes it would be better to change only the SCD associated with the sentence.

6.4. Interim Conclusion

FrESH solves only one part of the Problem II (Subsection 3.2.3) we came across building the SCD-based IR agent. Humans indicate faulty sentences, which then can be removed from the matrix without retraining the matrix using FrESH. However, FrESH removes a faulty sentence entirely from the SCD matrix and corpus. Thus, FrESH allows human users of the SCD-based IR agent to remove faulty content, i.e., *Fake-News* and copyright or privacy protected content.

However, dependencies and relations to other SCDs and other sentences are not maintained when a sentence is removed. Additionally, often the content of a sentence is correct, but the association between sentence and SCD is erroneous. The next chapter focuses on maintaining these dependencies without removing a sentence from the corpus.

7. ReFrESH: Relation-preserving Feedback-reliant Enhancement of SCDs

The sections of this chapter are largely taken verbatim from:

- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **ReFrESH – Relation-preserving Feedback-reliant Enhancement of Subjective Content Descriptions** in *18th IEEE International Conference on Semantic Computing (ICSC 2024)* – Best Paper Award
<https://dx.doi.org/10.1109/ICSC59802.2024.00010>

Magnus Bender developed the initial idea, conducted the experiments, and wrote the manuscript of the publication. The other three authors fundamentally supervised the research by discussing ideas, proofreading the manuscript multiple times, and giving feedback.

7.1. Introduction

FrESH removes faulty sentences and their SCDs entirely from a corpus. However, removing the entire sentence does not solve the problem this chapter addresses: A faulty association between an SCD and a sentence needs to be updated based on feedback from a user, but the sentence needs to remain in the corpus. Thus, this chapter provides ReFrESH, the solution to the second part of Problem II (Subsection 3.2.3). ReFrESH is an approach for Relation-preserving Feedback-reliant Enhancement of SCDs by Humans. Both, ReFrESH and FrESH incrementally change SCD-based models based on feedback.

Notably, correcting faulty associations between sentences and SCDs is required more frequently than removing sentences: A human or an automated annotation technique, e.g., OpenIE [AJPM15] or USEM, may create SCDs for a specific corpus. In general, when reading a text document, each human gets its own perceptions and views of the text document. For example, think again about studying for an exam. You take the script and start to annotate things you consider crucial with your understanding. Consequently, the SCDs added to a corpus by a human are slightly different and *subjective* depending on the particular human. Similarly, SCDs estimated with automated annotation techniques depend on the particular technique.

We assume that we have a corpus associated with SCDs to start with. Together, the text documents and the associated SCDs build a model of the corpus. An SCD-based IR agent answers queries of users which may be humans or other agents. A query is some unseen text to which the user is interested in finding similar and relevant documents from the agent’s corpus. To answer a query, the IR agent uses the SCDs associated with the corpus and returns documents that are assumed to be similar because they share the same SCDs as the query. A user may respond with feedback on the IR agent’s answer, i.e., may report a faulty or not similar document.

The IR agent heavily relies on SCDs. However, in most cases, a user of the agent will not create the initial SCDs of the corpus itself. At this point, there may be a difference between the SCDs used by the agent and the SCDs envisioned by the user. In other words, the SCDs used for IR do not necessarily represent the perceptions of the agent’s user. One possibility to avoid the difference is to force each user to create the initial SCDs of the corpus itself. However, having each user annotate the whole corpus with its understanding is not practical. Therefore, we update the model in case the user determines a mismatch. Then, the IR agents can change the SCDs using ReFrESH according to the feedback about the mismatch.

ReFrESH assigns the sentence with the faulty association to a different and hopefully better fitting SCD. The other sentences of the SCD are also considered and may be assigned to a different, a new, or the same SCD. Considering all sentences is necessary, as the creation of SCDs consists of multiple steps whereas each step influences the next steps. However, influences between steps cannot be reproduced retrospectively and thus can not easily be considered by ReFrESH. The term relation-preserving emphasizes that relations among SCDs and other sentences associated with SCDs are considered by ReFrESH. A relation, e.g., *homonym*, between two SCDs, one about a river bank and one about a financial institution, is preserved.

The remainder of this chapter is structured as follows: First, we look at related work. Afterwards, we formalize the problem of updating a single SCD while preserving relations to other SCDs and sentences and present the solution ReFrESH. Finally, we present an evaluation of ReFrESH and conclude afterwards.

7.2. Related Work

Before we present ReFrESH, we take a look at related work. Incrementally updating or changing already available models has been investigated in different ways, but not with SCDs (besides of FrESH).

One well-known approach is to pre-train a more general model first and fine-tune it later for a specific task. During fine-tuning the model is trained on task-specific

data. A typical example are models that use the transformer architecture like BERT [DCLT19] or GPT [BMR⁺20].

Another possibility is to bring the task-specific data in during the computation of the answer. In this case, the model is not updated, but to each query some user- and case-specific data is added before the query is processed by the model [LPP⁺20]. Most chatbots like ChatGPT¹ or Gemini² use this technique.

Both of these techniques incrementally update a model, like ReFrESH, and work, unlike ReFrESH, with models based on deep learning. Techniques that update a model must be distinguished from techniques that completely remove an item from the model, such as FrESH. There are approaches to remove an item from a model, e.g., for K-Means [Llo82, GGVZ19] or linear and logistic regression [ISCZ21]. The common idea is to avoid retraining the model and instead only incrementally change the model by applying an inverse operation that removes a single item of the training data from the model.

In this chapter, we focus on incrementally updating faulty associations between SCDs and sentences, while leaving the corpus unchanged and preserving all sentences.

7.3. Relation-preserving Updates on SCD Matrices

This section introduces ReFrESH, the algorithm that allows for updating an SCD-based model by correcting faulty associations between sentences and SCDs. First, we look at possible relations among SCDs and describe ReFrESH afterwards.

7.3.1. Relations to Preserve

Before we can compose a relation-preserving algorithm, we need to specify the different relations among SCDs. An SCD t_i consists of referenced sentences $\{s_1, \dots, s_S\}$, a word distribution $(v_{i,1}, \dots, v_{i,L})$, and additional data \mathcal{C}_i , these are the intra-SCD relations. One of the referenced sentences s_r has been marked as faulty and should be removed. However, the remaining $S - 1$ sentences $\mathcal{S}_c = \{s_1, \dots, s_S\} \setminus \{s_r\}$ are then considered as correct. The word distribution will be different after a sentence is removed from an SCD, but can be easily recalculated afterwards. All items of the additional data \mathcal{C}_i are related, i.e., each item can be understood as related to its SCD and thus as a relation of the SCD to be preserved.

¹<https://chat.openai.com/>

²<https://gemini.google.com/>, formerly Bard

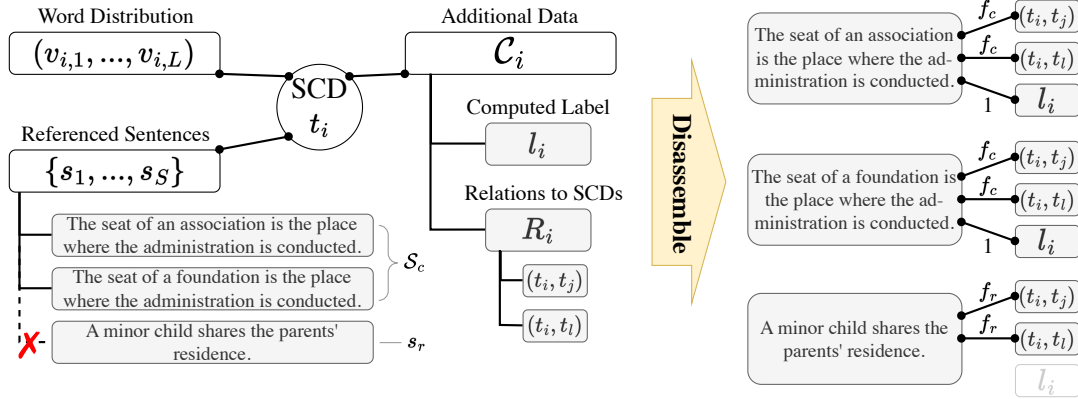


Figure 7.1.: Left: An SCD with its referenced sentences, three in this example, its row of the word distribution, and its additional data, e.g., containing a label and two relations to other SCDs. The red cross marks the sentence to remove from the SCD. Right: The SCD after the disassemble step. Each of the three sentences now stand by itself, while the label and the two relations have been reassigned to the individual sentences and given a factor.

Summarized, ReFrESH needs to preserve the relations of the referenced sentences s_r and \mathcal{S}_c to the items in \mathcal{C}_i . In the following, \mathcal{C}_i contains a label l_i , e.g, computed by LESS, and a set of relations to other SCDs R_i , i.e., inter-SCD relations. Each tuple in R_i models a relation of SCDs, e.g., (t_i, t_j) a relation between t_i and t_j .

In general, a sentence may have relations which directly belong to the sentence itself and not to its SCD. As ReFrESH only modifies SCDs, relations belonging to a sentence are not effected by ReFrESH. ReFrESH uses this by shifting relations from the SCD to the sentences. After a relation is shifted to the sentences, ReFrESH does not have to distinguish between intra- and inter-SCD relations.

On the left hand side of Figure 7.1, an SCD with the previously described parts is depicted. In this example, the SCD references three sentences. The sentence with the faulty association is already marked by a red cross. The additional data contains a label and two relations of t_i to other SCDs, i.e., t_j and t_l .

7.3.2. Four Steps for Updating an SCD

ReFrESH needs to preserve the relations between the referenced sentences and the items in the additional data of an SCD. After applying ReFrESH, the sentence s_r and the sentences \mathcal{S}_c are associated with different SCDs.

We assume that there is a corpus \mathcal{D} with an SCD matrix $\delta(\mathcal{D})$ and the set of SCDs $g(\mathcal{D})$. Together, these three parts build an SCD-based model which is updated by ReFrESH. To do so, ReFrESH needs four steps:

1) Shift Relations to Sentences

The input of ReFrESH is a sentence s_r which is falsely associated with an SCD t_i . ReFrESH's task is to remove s_r from t_i and to reassign s_r to a better fitting SCD. Besides preserving the relations, ReFrESH also needs to consider the impact of s_r on the SCD t_i . When s_r is associated with t_i , the word distribution of t_i also represents s_r . Thus, similar sentences to s_r might also be added to t_i , just because s_r is associated with t_i . This is based on the assumption that sentences are added to an SCD one after another and using the word distribution to measure similarity. For example, USEM chooses best matching sentences in a greedy way one after one. Also, humans may build their own set of SCDs manually and incrementally (including the use of ReFrESH multiple times in a row). Thus, when s_r is removed, other sentences added because of s_r may also need to be removed from the SCD.

A comparable scenario arises when clustering data points using a density based clustering algorithm: If a point at a border of a cluster is sufficiently close to another cluster, both clusters may get merged. In this case, the point at the border becomes some type of bridge between both clusters and without this point both clusters would not have been merged. Hence, if this border point is removed later, both clusters should be separated, too.

To take such influences into account, ReFrESH needs to consider all referenced sentences $\{s_1, \dots, s_S\}$ of t_i . To be able to consider every sentence individually, ReFrESH first shifts all the relations to preserve, the ones in \mathcal{C}_i , directly to the individual sentences. A sentence to be removed might not share the topic of an SCD, thus, ReFrESH does not shift the label to s_r . However, the other sentences \mathcal{S}_c , which are considered correct, will share the topic of the SCD. Thus, the sentences in \mathcal{S}_c are assigned with the label l_i and the factor 1 for this relation between sentence and label.

Next, ReFrESH does the shift for all the relations in R_i and again differentiates between sentences s_r and \mathcal{S}_c . The relations in R_i are added to \mathcal{S}_c with the factor f_c for correct sentences. Thereby, f_c is defined as $f_c = \frac{S-1}{S}$ where S is the number of referenced sentences of t_i . In contrast, the relations in R_i are added to s_r with the factor f_r for removed sentences. Here, the factor is $f_r = \frac{1}{S}$. If some relation already has a factor, both factors are multiplied because the factors express uncertainty.

The relations from \mathcal{C}_i are shifted to the sentences to make sure all relations are preserved. The factors make sure that relations are preserved differently for \mathcal{S}_c and

s_r , i.e., if a sentence is considered correct, the relations of the SCD are also more likely to be correct than if the sentence is falsely associated. Of course, it might be necessary to change the factor for specific relations and use-cases. All sentences and relations are stored in \mathcal{P} for later use.

2) Disassemble SCD

Coming back to the problem that we do not know why a sentence was added to an SCD. ReFrESH cannot determine which sentences have been added because of s_r , too. The only solution to this problem is to disassemble the entire SCD t_i . After step 1), all relations to preserve are directly tied to each referenced sentence. Thus, the SCD t_i is not needed anymore and can be disassembled without losing any important information. Afterwards, ReFrESH is able to consider each sentence separately.

Disassembling an SCD means deleting the word distribution $(v_{i,1}, \dots, v_{i,L})$, the i -th row, from the SCD matrix and removing t_i from $g(\mathcal{D})$. Additionally, the SCDs in relation with t_i are updated that they are now in relation with the referenced sentences of t_i .

On the right hand side of Figure 7.1, an example of an disassembled SCD t_i is shown. The lowest sentence is s_r , in this case the factors are $f_r = \frac{1}{3}$ and $f_c = \frac{2}{3}$. The label gets a factor of 1 for the upper two sentences \mathcal{S}_c and s_r has no label. The SCD t_i is temporarily removed completely.

3) Reassign Sentences to SCDs

Now, the previously referenced sentences stored in \mathcal{P} need to be reassigned, i.e., each sentence needs to be associated with a new SCD. In this third step, ReFrESH needs to find the best SCD for each sentence. This best SCD may be an already known SCD of the corpus or newly composed SCDs while ReFrESH needs to assure that s_r does not get associated with a new SCD. A new SCD references only sentences from \mathcal{S}_c . If s_r gets associated with such new SCD, the association of s_r and the new SCD might be very similar or even equal to the initial SCD t_i before running ReFrESH to remove s_r .

Analogously, in the example about the clustering of data points, all points of both clusters and the border point would be considered again. Each point may be added to another cluster or one or more new clusters may be created, while the border point will become an outlier or member of another cluster.

Algorithm 9 Relation-preserving Feedback-reliant Enhancement of SCDs

```
1: function REFRESH( $(\mathcal{D}, \delta(\mathcal{D}), g(\mathcal{D})), s_r, t_i$ )
2:   Input: SCD-based model  $(\mathcal{D}, \delta(\mathcal{D}), g(\mathcal{D}))$ ;
3:   sentence to remove  $s_r$ ; associated SCD  $t_i$ 
4:   Output: Updated model  $(\mathcal{D}, \delta(\mathcal{D}), g(\mathcal{D}))$ 
5:    $\mathcal{P} \leftarrow \emptyset$  ▷ Sentences with relations to preserve
6:   for each referenced sentence  $s_i \in \{s_1, \dots, s_S\}$  do
7:     if  $s_i = s_r$  then ▷ Calculate factor
8:        $f \leftarrow \frac{1}{S}, p_i \leftarrow \emptyset$ 
9:     else ▷ Preserve label for sentences in  $\mathcal{S}_c$ 
10:       $f \leftarrow \frac{S-1}{S}, p_i \leftarrow \{(1, l_i)\}$ 
11:    for each relation to preserve  $r_i \in R_i$  do
12:       $p_i \leftarrow p_i \cup \{(f, r_i)\}$  ▷ Store with factor
13:     $\mathcal{P} \leftarrow \mathcal{P} \cup \{(s_i, p_i)\}$  ▷ Store sentence and relation
14:    ▷ Step 1) Shift Relations to Sentences
15:     $g(\mathcal{D}) \leftarrow g(\mathcal{D}) \setminus t_i$ 
16:     $\delta(\mathcal{D})[i] \leftarrow Nil$  ▷ Delete  $i$ -th row
17:    ▷ Step 2) Disassemble SCD
18:     $\mathcal{N} \leftarrow \emptyset$  ▷ Note changed SCDs
19:    ▷ First reassign  $s_r$ 
20:     $j \leftarrow \text{MOSTSIMILARROW}(\vec{s}_r, \delta(\mathcal{D}))$ 
21:     $\delta(\mathcal{D})[j] \leftarrow \delta(\mathcal{D})[j] + \vec{s}_r$  ▷ Update matrix
22:     $t_j \leftarrow (\mathcal{C}_j, \{s_1, \dots, s_S\} \cup \{s_r\})$  ▷ Add  $s_r$  to  $t_j$ 
23:     $\mathcal{N} \leftarrow \mathcal{N} \cup \{(t_j, s_r, p_r)\}$  ▷ Note that  $t_j$  changed
24:    ▷ Reassign remaining sentences  $\mathcal{S}_c$ 
25:    for each sentence and rel.  $(s_i, p_i) \in \mathcal{P} \setminus (s_r, p_r)$  do
26:       $j \leftarrow \text{MOSTSIMILAR}(\vec{s}_i, \delta(\mathcal{D}))$ 
27:       $k \leftarrow \text{MOSTSIMILAR}(\vec{s}_i, \vec{\mathcal{P}} \setminus \vec{s}_i)$ 
28:      if  $\text{SIMILARITY}(j) > \text{SIMILARITY}(k)$  then
29:         $\delta(\mathcal{D})[j] \leftarrow \delta(\mathcal{D})[j] + \vec{s}_i$  ▷ Update matrix
30:         $t_j \leftarrow (\mathcal{C}_j, \{s_1, \dots, s_S\} \cup \{s_i\})$  ▷ Add  $s_i$  to  $t_j$ 
31:         $\mathcal{N} \leftarrow \mathcal{N} \cup \{(t_j, s_i, p_i)\}$ 
32:      else ▷ Build new SCD  $t_k$  with  $s_i$  and  $s_k$ 
33:         $\delta(\mathcal{D})[k] \leftarrow \vec{s}_i + \vec{s}_k$  ▷ Add row to matrix
34:         $g(\mathcal{D}) \leftarrow g(\mathcal{D}) \cup (\mathcal{C}_k, \{s_i, s_k\})$  ▷ Create SCD
35:         $\mathcal{P} \leftarrow \mathcal{P} \setminus (s_k, p_k)$  ▷  $s_k$  already reassigned
36:         $\mathcal{N} \leftarrow \mathcal{N} \cup \{(t_k, s_i, p_i), (t_k, s_k, p_k)\}$ 
37:        ▷ Step 4) Propagate new Relations
38:    for each SCD, sentence, and rel.  $(t_i, s_i, p_i) \in \mathcal{N}$  do
39:      if  $\mathcal{C}_i = \emptyset$  then ▷ New SCD
40:         $\mathcal{C}_i \leftarrow p_i$ 
41:         $l_i = \text{LESS}(t_i)$ 
42:      else
43:        for each factor and relation  $(f_i, r_i) \in p_i$  do
44:           $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \left\{ \left( \frac{x}{S-x} \cdot f_i, r_i \right) \right\}$ 
45:    return  $(\mathcal{D}, \delta(\mathcal{D}), g(\mathcal{D}))$ 
```

We still need an algorithm to reassign sentences to SCDs, which is similar to estimating SCDs in an unsupervised way. Thus, ReFrESH applies the idea of USEM and uses USEM’s greedy method to reassign the sentences with new or known SCDs.

The idea of USEM is to start by considering each sentence as an SCD with one referenced sentence. Afterwards, USEM uses its greedy method and identifies the two most similar SCDs to merge. Hence, in the first iteration of USEM two SCDs with one referenced sentence each get merged and become one SCD with two referenced sentences. To identify similar SCDs, the cosine similarity is used with the SCD matrix’ rows.

For ReFrESH, the idea is applied as follows: First, the word vector for s_r is calculated (as in Lines 8 and 9 of Algorithm 1) and s_r is added to most similar and already known SCD of the corpus (Lines 17 and 18 in Algorithm 9). For each sentence in \mathcal{S}_c its word vector is then compared to all rows in the SCD matrix and to the word vectors of the other sentences (Lines 22 and 23 in Algorithm 9). If a word vector is most similar to one of the rows in the SCD matrix, and thus to an already known SCD of the corpus, the sentence is added to the SCD as referenced sentence (Lines 25 - 27 in Algorithm 9). In the other case, if a word vector is most similar to a word vector of another sentence, both sentences are merged to form a new SCD, which is then added to $g(\mathcal{D})$ and $\delta(\mathcal{D})$ (Lines 29 - 32 in Algorithm 9). This is repeated until all sentences of \mathcal{S}_c are part of an SCD. In the end of step 3), ReFrESH recalculates the word distribution of all SCDs to which new referenced sentences have been added. The new and modified SCDs are stored in \mathcal{N} .

4) Propagate new Relations

Finally, the SCD-based model contains all sentences again and all SCDs in the model have their word distribution and set of referenced sentences. However, (i) there is no additional data of all new SCDs and (ii) the SCDs that have received one or more new referenced sentences do not have the relations of their new sentences. Algorithm 9 iterates through all modified SCDs in \mathcal{N} including the sentences and relations and addresses both cases.

In the case of (i), the new SCD does not have any relations itself. Furthermore, the relations of the referenced sentences of this SCD are all the same, as all sentences originate from the same disassembled SCD. Thus, the relations can be shifted back from the sentences to the SCD including the factors. The factors outline some uncertainties, as relations may not originally originate from the SCD and its sentences. Finally, a new label for the SCD is generated by LESS (Chapter 5).

Otherwise, case (ii), we assume that x new sentences have been added to an SCD with previously S sentences. In contrast to (i), the relations stay with the sentences

and are also propagated to the SCD. The relations from the sentences are added to the SCD's additional data and each factor is multiplied by $\frac{x}{S+x}$. By using this factor, each relation is weighted depending the ratio it has among the referenced sentences of the SCD. The label of the SCD will not be changed, but labels from the sentences are added like a relation including the factor. The factors used with the relations by ReFrESH are slightly inspired by weighted model counting [SBK05].

Finally, the original association of s_r with t_i has been removed and all former referenced sentences of t_i have been reassigned to a new and better fitting SCD. All relations have been preserved and propagated to other SCDs, too.

7.3.3. Algorithm ReFrESH

Based on the previous subsection presenting the four steps of ReFrESH, it is entirely formulated in Algorithm 9. ReFrESH follows the four steps and returns the updated SCD-based model as triple $(\mathcal{D}, \delta(\mathcal{D}), g(\mathcal{D}))$. The input contains the sentence to remove s_r and its SCD t_i . SCD t_i might be omitted. Then all SCDs in $g(\mathcal{D})$ must be searched for the SCD associated with s_r .

We have now proposed the algorithm of ReFrESH with four steps. Next, we describe the evaluation and discuss the results.

7.4. Evaluation

After we have introduced ReFrESH, we present an evaluation. First, we introduce the corpus. Afterwards, we describe the workflow of the evaluation and the used metrics. Finally, we present the results and discuss the performance of ReFrESH.

7.4.1. Dataset

In this evaluation we again use the Bürgerliches Gesetzbuch (BGB)³, the Civil Code of Germany, in German language as corpus. The BGB is freely available, can be downloaded as XML file, and it is easily parsable and processable. As the corpus is a law text it consists of correct language, i.e., punctuation and spelling follow the orthographic rules. Thus, less preprocessing and no data cleaning is needed. Furthermore, the words used in text documents have a clear meaning and mostly the same words are used instead of using synonyms.

³<https://www.gesetze-im-internet.de/bgb/>, English translation https://www.gesetze-im-internet.de/englisch_bgb/

We use the first part of the BGB, the so called “General Part”: The entire corpus consists of 228 law paragraphs and overall 854 sentences which are used as SCD windows. Each law paragraph contains between 1 and 40 sentences with an average of 3.78 sentences. The vocabulary consists of 1436 words, where each sentence is between 1 and 20 words long with an average of 7.11 words.

7.4.2. Workflow and Implementation

ReFrESH is implemented using Python and runs inside a Docker container. The implementation uses the libraries Gensim⁴, NumPy⁵, and NLTK⁶. The evaluation is performed on a machine featuring 8 Intel 6248 cores at 2.50GHz (up to 3.90GHz) and 16GB RAM. We run the following workflow to evaluate ReFrESH:

- (i) Randomly choose a set of pairs of sentences which do not share a similar concept, the set contains around one eighth of all the sentences of the corpus. Each pair of sentences is associated with the same SCD and then acts as faulty associations of SCD and sentences.
- (ii) Estimate an SCD-based model which contains the faulty associations chosen in (i): Use USEM with the greedy method and estimate the *faulty* SCD matrix $\delta_f(\mathcal{D})$. We add a step to USEM after the initial SCD matrix is created. This step groups each pair of sentences from (i) into the same SCD, which leads to faulty associations in the model. Afterwards, USEM continues normally, i.e., finds similar sentences and groups them into SCDs.
- (iii) Run ReFrESH to update $\delta_f(\mathcal{D})$ and remove all the faulty associations initiated by the pairs of sentences from (i). Meanwhile, keep a copy of $\delta_f(\mathcal{D})$ and create the new *refreshed* SCD matrix $\delta_r(\mathcal{D})$, where all the faulty associations have been removed.
- (iv) Create a baseline model which represents the correct model for the corpus \mathcal{D} . Estimate the *baseline* SCD matrix $\delta_b(\mathcal{D})$ using USEM without the additional step.
- (v) Compare the differences between the three models, i.e., the matrices $\delta_f(\mathcal{D})$, $\delta_r(\mathcal{D})$, and $\delta_b(\mathcal{D})$.

This workflow focuses on evaluating step two (disassemble) and step three (reassign). The reassignment of sentences to new and better SCDs is the crucial and approximate part of ReFrESH. It is important to maintain the relations of the SCDs and

⁴<https://radimrehurek.com/gensim/>

⁵<https://numpy.org/>

⁶<https://www.nltk.org/>

sentences (steps one and four), but their treatment is fixed by the algorithm and not approximate. Hence, steps one and four do not need to be evaluated.

7.4.3. Metrics

Based on the three matrices $\delta_f(\mathcal{D})$, $\delta_r(\mathcal{D})$, and $\delta_b(\mathcal{D})$ we need to evaluate the performance of ReFrESH. The main idea is that the distributions of our baseline $\delta_b(\mathcal{D})$ and the refreshed $\delta_r(\mathcal{D})$ should be identical. For $\delta_r(\mathcal{D})$ first some faulty associations have been added and removed afterwards by ReFrESH, while $\delta_b(\mathcal{D})$ is trained straightforward. Thus, we need to measure the difference between matrices of distributions.

Using the Hellinger distance [Hel09], the distance between two matrices P and Q can be calculated row-wise by:

$$h_i(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{j=1}^L \left(\sqrt{P[i][j]} - \sqrt{Q[i][j]} \right)^2}$$

The resulting distance vector $H(P, Q)$ contains in each row $h_i(P, Q)$ the distances between the matrix' rows. Based on this distance vector, we calculate two metrics:

First, the proportion of differences, which is the proportion of rows in H which are not equal to zero. It shows how many SCDs are different between two SCD matrices. Since ReFrESH updates the SCD matrix, we assume that there are many equal, i.e. unchanged, rows. Second, the average Hellinger distance considers only the non equal rows in H and represents the average difference in H . It shows how similar the SCDs of two SCD matrices are.

A technical note: The distances cannot be calculated between two SCD matrices directly, because the row numbers of an SCD matrix might change between multiple runs of USEM or ReFrESH. Thus, we first create intermediate matrices which use a globally equal window number and calculate the distances on these intermediate matrices. Using this window numbers, we compare the distributions for the same sentence between different matrices.

7.4.4. Results

In this section, we present the results gained using ReFrESH and the previously described workflow. In the upper graph of Figure 7.2, the average Hellinger distance is shown for different numbers of SCDs. Each number of SCDs represents one run of the workflow and thus three matrices *faulty*, *baseline*, and *refreshed*. Using the

7. ReFrESH – Relation-preserving Feedback-reliant Enhancement of SCDs

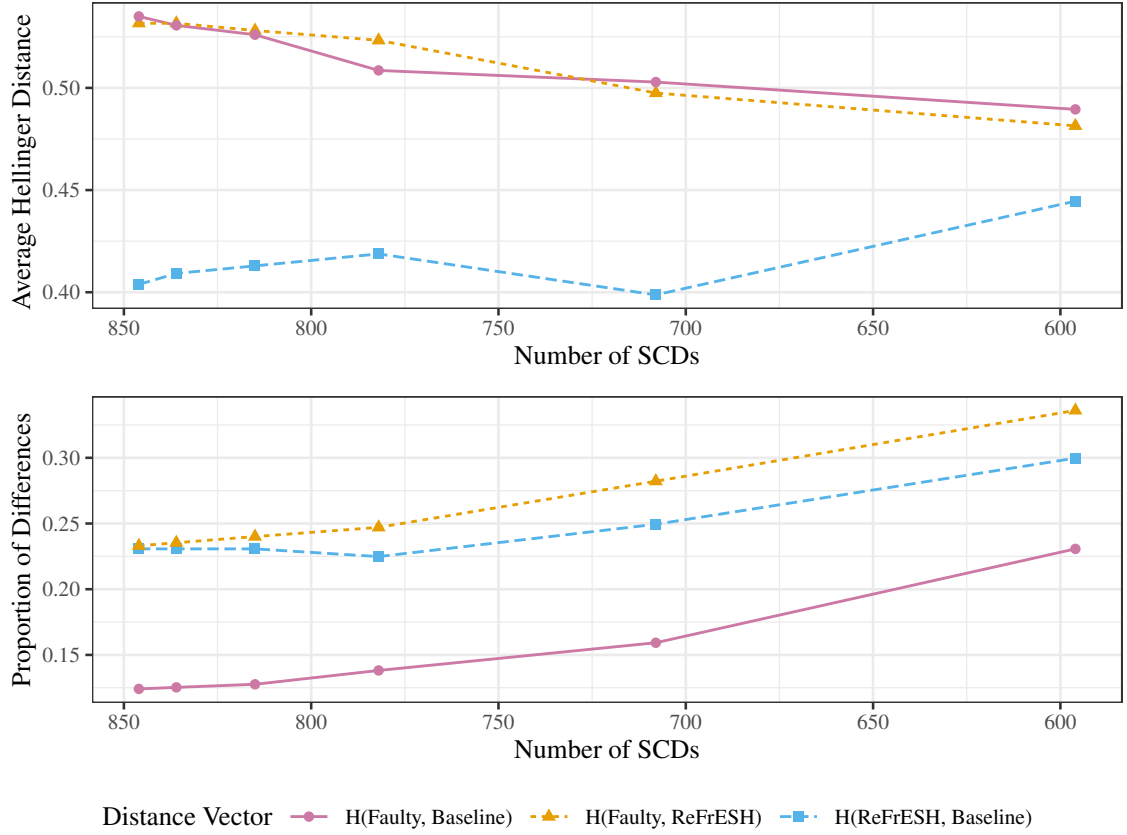


Figure 7.2.: Proportion of different rows and average Hellinger distance value for multiple hyperparameters of USEM resulting in different numbers of SCDs. For each number of SCDs, three distances, each for one of the distance vectors between the three generated matrices, are shown.

three matrices we calculate three Hellinger distance vectors $H(\text{Faulty}, \text{Baseline})$, $H(\text{Faulty}, \text{ReFrESH})$, and $H(\text{ReFrESH}, \text{Baseline})$ and for each vector both metrics.

The performance of ReFrESH is especially shown by the dashed blue line of $H(\text{ReFrESH}, \text{Baseline})$. A perfect deletion of faulty associations from the SCD matrix would result in a distance of zero. In this case, the distance is greater than zero, but well below the other two lines. The solid purple line shows $H(\text{Faulty}, \text{Baseline})$, which can be interpreted as the *error* an SCD-based model would have without ReFrESH, because a faulty matrix with all the faulty associations is compared to the baseline. Since the dashed blue line is below the solid purple line, ReFrESH reduces the *error* of the model by removing faulty associations.

In the lower graph of Figure 7.2, the proportions of different rows are shown—again for different numbers of SCDs and based on three Hellinger distance vectors. The two dashed lines represent a distance to *ReFrESH* and are above the solid purple line of $H(\textit{Faulty}, \textit{Baseline})$. A smaller amount of different rows in $H(\textit{Faulty}, \textit{Baseline})$ may be explained by the fact that both models use USEM. In contrast, *ReFrESH* is a different technique and reassigns sentences to other SCDs which in total affects more SCDs.

In Figure 7.3, the reduction of the Hellinger distance by running *ReFrESH* is shown. It shows the average difference of $H(\textit{Faulty}, \textit{Baseline})$ and $H(\textit{ReFrESH}, \textit{Baseline})$, in other words, the space between the dashed blue and solid purple line in the upper part of Figure 7.2. Hence, the value can be seen as an improvement of the model when using *ReFrESH*.

At first glance, the improvement might be a bit small. However, *ReFrESH* leads to matrices with more different rows but with smaller distances of each row. In comparison, the baseline and the faulty model share some identical rows, with each distance value being significantly larger. *ReFrESH* does the reassignment of the sentences to SCDs with the goal of finding a better matching SCD. To do so, *ReFrESH* needs to change many SCDs with the goal of getting a slightly changed but better model. Summarized, *ReFrESH* provides a good performance for refreshing an SCD-based model based on, e.g., human, feedback.

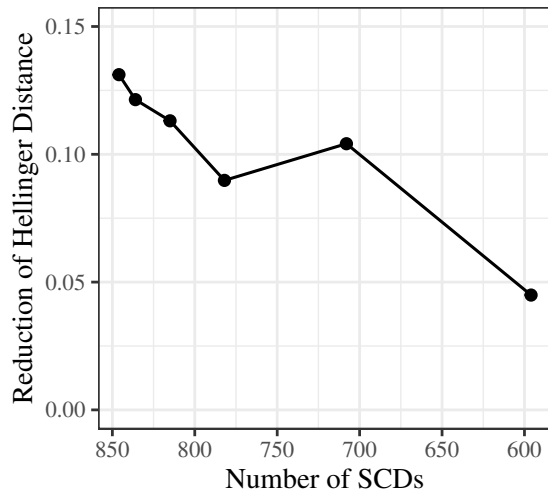


Figure 7.3.: Reduction of the Hellinger distance when running *ReFrESH* on the faulty model and comparing to the baseline, i.e., the amount of correction done by *ReFrESH*.

7.5. Interim Conclusion

This chapter introduces *ReFrESH*, an approach consisting of four steps for incorporating feedback in SCD-based models. In general, when reading a text document, each human gets its own perceptions and views of the text document. Hence, SCDs are slightly different, i.e., *subjective*, depending on the human or automated annotation technique used to create them. If SCDs are used by an IR agent, a user may

consider some answers of the agent faulty and respond with feedback to the agent. In this case, the agent can use ReFrESH to update its SCDs. It allows incremental updates of SCD-based models based on user feedback and avoids the need for each user to create their own SCDs for each corpus from scratch.

In the first step, ReFrESH shifts all relations among SCDs to the sentences to ensure that relations between SCDs are preserved during the update. Second, the SCD to be updated is disassembled before each sentence is reassigned to a better fitting SCD in the third step. Finally, the preserved relations are propagated back from the sentences to the SCDs.

Overall, the evaluation shows that ReFrESH works well and provides a powerful technique to update SCD-based models based on human feedback. Using ReFrESH, faulty associations between sentences and SCDs can be removed and the sentences get associated with new and better fitting SCDs. The evaluation focuses on step two (disassemble) and step three (reassign) because the crucial and approximative part of ReFrESH is the reassignment of an SCD.

Closing the Cycle For the goal of this dissertation and for creating an SCD-based IR agent, ReFrESH is an elementary piece of the puzzle. Combining all the previous chapters of this dissertation, we can close the cycle of estimation, enrichment, usage, and improvement approaching text understanding. USEM and LESS provide the initial set of SCDs and the SCD matrix. If there are already some SCDs, SEM may be used together with some of the strategies requiring less supervision by Kuhr et al. [KWM19, KBBM21, BBG⁺21b, BBG⁺21a]. Afterwards, the MPS²CD algorithm is used for IR and the corpus can be maintained and extended with relevant documents [KBBM20]. Finally, the cycle gets closed by FrESH and ReFrESH: Both techniques update the SCDs by feedback and thus improve the SCDs which are then again used by the IR agent.

However, our Problem IV (Subsection 3.2.5) remains: ReFrESH introduces factor-weighted relations between SCDs and the sentences of a corpus. Thus, the next chapter introduces relations among SCDs using complementarity as an example. Looking back at Subsection 5.2.1, these relations between SCDs are considered as inter-SCD relations.

8. Complementarity as an Inter-SCD Relation

This chapter is based on the following two publications:

- Magnus Bender, Felix Kuhr, Tanya Braun: **To Extend or not to Extend? Enriching a Corpus with Complementary and Related Documents** in *International Journal of Semantic Computing*, 2022
<https://dx.doi.org/10.1142/S1793351X2240013X>
- Magnus Bender, Felix Kuhr, Tanya Braun: **To Extend or not to Extend? Complementary Documents** in *16th IEEE International Conference on Semantic Computing (ICSC 2022)*
<https://dx.doi.org/10.1109/ICSC52841.2022.00011>

For the conference paper: All three authors developed the idea. Magnus Bender and Felix Kuhr wrote the manuscript. Magnus Bender conducted the experiments. Tanya Braun fundamentally supervised the research.

For the journal article: The journal article itself is an extended version of the conference paper with additional major changes to the structure and use-case. Magnus Bender developed the initial idea, conducted the experiments, and wrote the manuscript. Tanya Braun fundamentally supervised the research by discussing ideas, proofreading and lecturing the manuscript, and giving feedback. The actual sections of this chapter are largely taken verbatim from the journal article.

8.1. Introduction

In this chapter, we consider inter-SCD relations, i.e., relations between two SCDs. We use the example of complementarity as relation and introduce techniques for working with complementary SCDs, which can be generalized for other types of inter-SCD relations.

Still, we follow the idea of building an SCD-based IR agent. On one hand, relations can be added by the users of the IR agent. Relations are then used to generate responses. On the other hand, the IR agent needs to maintain its corpus, i.e., searches for new documents to extend its corpus, e.g., to add new information and

8. Complementarity as an Inter-SCD Relation

provide a versatile collection of documents in the responses. We refer to this internal task of an agent as corpus extension.

To decide a corpus extension, the agent has to determine if a document is relevant to a corpus. The first idea is that related documents are relevant. Relatedness can be captured by some measure of similarity, defined using words directly or representations derived from them such as topic-word probability distributions, inferring abstract topics represented as distributions over a vocabulary, or SCD-word probability distributions. Kuhr et al. [KBBM19, KBBM20] have worked with four document categories based on similarity using SCD-word probability distributions: (i) quasi copies, a.k.a. similar documents, (ii) extensions, (iii) revisions, and (iv) unrelated documents. Classifying documents on similarity may lead to looking at documents that only contain more of the same, albeit possibly updated information. Hence, the agent needs further techniques not using similarity to identify other relevant documents.

To be able to leave the bubble of similarity, we need to define a different measure of relatedness. Therefore, we focus on adding a fifth document category *complement*, which is hard to define given only words or numbers in distributions or vector representations. Complements may use a completely different vocabulary, which may render it as an unrelated document given similarity measures based on words. In terms of vector representations, one may think of a complement as a document having high values in certain dimensions where another one has low values. This consideration may also apply to completely unrelated documents, though, making it a not very effective measure. Therefore, we consider two problems, (i) formally defining complementary documents and distinguishing complementary documents from unrelated documents, and (ii) extending the document classification technique by Kuhr et al. [KBBM19, KBBM20] with a fifth document category *complement*.

To get a handle on complementarity by way of a formal definition, we turn to SCDs, specifically, SCDs in the form of relational tuples such as *spo*-triples together with a taxonomy that specifies a concept hierarchy for the constants occurring in SCDs. We hypothesize the following: Complementary documents have SCDs that contain different constants of the same concept in a taxonomy. Given this hypothesis, we can formally define complementary documents and specify a corresponding document classification problem. Given a definition of complementarity, we can solve the first problem of distinguishing unrelated and complementary documents by calculating a complementarity value between two documents based on their SCDs and how they interrelate given a taxonomy.

Facing the second problem of extending the document classification technique by Kuhr et al. [KBBM19, KBBM20] with a category *complement*, we extend the SCD-word probability distribution with *complementary* SCDs originating from complementary documents. Thus, the combined SCD-word probability distribution con-

tains related and corresponding complementary SCDs. For a new document to classify, the agent is then able to estimate most probable related and complementary SCDs using the combined SCD-word distribution. Altogether, the agent uses the estimated SCDs to classify a document in one of the five categories.

Specifically, the contributions of this chapter are:

- (i) a definition of the document classification problem for complements and a definition of complementarity for SCDs in the form of relational tuples and a definition of complementarity for documents based on complementary SCDs,
- (ii) a first solution approach to the problem, which can be used for distinguishing unrelated and complementary documents,
- (iii) an altogether document classification technique for complementary and related documents based on Kuhr et al., and
- (iv) an evaluation of the performance of distinguishing unrelated and complementary documents as well as the overall document classification performance, comparing this chapter's approach against the method by Kuhr et al.

The remainder of this chapter is structured as follows: We start with related work followed by a recap of SCDs and document categories. Then, we specify complementarity based on SCDs and present a solution approach to identify complementary documents. Next, we present how to integrate complementary SCDs in the SCD matrix and how to classify documents using this distribution. Finally, we present an evaluation and end with an interim conclusion.

8.2. Related Work

To identify complementarity, we need to extract constants available in the taxonomy from sentences. Thus, we are interested in automatic (semantic) annotation systems which might be useful. Generally, these annotation systems attach additional data to various concepts, e.g., people, organizations, or places, in a given text, enriching the documents with machine-processable data. Some famous automatic annotation systems are YEDDA [YZLL18], Slate [Kum19], MINTE [CGR⁺17], and YAGO [SKW07]. For further annotation systems, please refer to [LZ19]. Some annotation systems like OpenCalais¹ automatically attach data from a database, e.g., DBpedia [ABK⁺07], to extractable Named Entities (NEs) in the text. That is, the extractable NEs are matched to items in a database and data from the database is added to the documents next to the NEs.

¹<http://www.opencalais.com/>, archived at <https://web.archive.org/web/20180902204730/http://www.opencalais.com/>

In this chapter, we investigate a different but related problem, namely estimating the complementarity of a new document with respect to the documents in a corpus of an agent. The complementarity of a document is based on NEs extractable from the text of the document and the NEs available in documents in the corpus. An agent can decide to extend a corpus with a new document complementary to documents in its corpus. In general, other automatic annotation systems ignore the context of a user and add data to documents already available in the corpus of an agent.

Surveying methods of text mining, one can base a decision if a new document provides a value for an agent on different aspects, e.g., (i) similarity of text in the spirit of tf.idf [SJ72], comparing a vector representation of a new document with vector representations of the documents in the corpus, (ii) similarity of topics in the spirit of LDA [BNJ03], comparing an estimated topic distribution of a new document with topic distributions of documents in a given corpus, or (iii) entity matching [NKAJ59] using NE recognition, comparing entities (and relations) retrieved from the new document with entities (and relations) from SCDs in the corpus. We aim at providing an approach to estimating the complementarity of a new document using a given concept hierarchy and entities. The first two approaches have drawbacks regarding identifying document categories including complementary documents: Both are bag-of-words approaches, i.e., they ignore the order of words and extractable NEs, while the order is required to identify extensions and revisions. Thus, we use elements from entity matching to link entities from a document to an external concept hierarchy.

Another class of related work deals with Hidden Markov Model (HMM)-based classification. Classification and statistical learning using HMMs has achieved remarkable progress in the past decades. Using an HMM is a well-investigated stochastic approach for modeling sequential data, and the generation process of HMMs has been successfully applied in a variety of fields, such as speech recognition [RJ86], character recognition [HBT96], finance data prediction [Zha04, NN15], credit card fraud detection [SKSM08], and workflow mining [LKM16]. An approach is to learn an HMM by the Baum-Welch algorithm [BPSW70], which is a special case of the EM algorithm [DLR77]. HMMs can be used for estimating the most likely sequence of hidden states in a dynamic programming fashion by the Viterbi algorithm [Vit67].

8.3. Preliminaries

This section specifies advanced notations for this chapter and recaps how an SCD matrix can be used to classify documents.

Generally, an SCD t_i is a tuple of additional data \mathcal{C}_i and the referenced sentences. In this chapter, we concretize this definition in a way that an SCD t_i is an *spo*-triple with references to the sentences having the *spo*-triple. Whereas s_i , p_i , and o_i represent the subject, predicate, and object. To comply with the general notation, the *spo*-triple might be added to the additional data \mathcal{C}_i .

The general SCD matrix $\delta(\mathcal{D})$ builds on a corpus of related documents \mathcal{D} . In this chapter, we have to distinguish between related and not related corpora. Thus, we call the general SCD matrix related SCD (rSCD) matrix $\delta_r(\mathcal{D}_r)$.

Next, we recap how to classify documents using MPS²CD and an rSCD matrix.

Corpus Extension using Similarity Using the MPS²CD similarity values, Kuhr et al. [KBBM19, KBBM20] present a method with which an agent can classify a new document d' by one of the following four categories:

- *Quasi copy* or *Related*: Document d' is classified as *sim* if the values in the MPS²CD similarity sequence are mostly high and contain only few entries with slightly lower values.
- *Extension*: Document d' is classified as *ext*, representing an extension of another document $d \in \mathcal{D}$, if d' is generated by *appending* a document d , i.e., d' represents an updated version of d .
- *Revision*: Document d' is classified as *rev*, representing a revision of another document $d \in \mathcal{D}$ generated by *replacing* or *removing* parts of d .
- *Unrelated document*: Document d' is classified as *unrel* if the values in the MPS²CD similarity sequence of d' are mostly low.

In the next section, we define complementarity of SCDs and documents leading to a new category *complement* of documents. Then, we describe an approach classifying documents of the new category *complement* based on the definitions.

8.4. Identifying Complementary Documents

This section presents an approach for identifying documents containing complementary content with respect to the content of documents in a given corpus. We use the task of corpus extension as the application scenario for complementary documents. However, the given definitions and algorithms can be adapted for other tasks such as document retrieval. First, we define a binary document classification problem, i.e., if a document is complementary. Second, we provide a definition of complementary documents based on SCD complementarity values to solve the problem.

Third, we present an approach to corpus extension with complementary documents by identifying a new document d' as a complement using the previously defined notions.

8.4.1. Document Classification Problem: Complement

Given an unknown document d' and a corpus \mathcal{D} , an agent might be interested in whether d' is a *complement* to documents in \mathcal{D} . Formally, we ask whether d' is a complement to d ($Complement = true$) or not ($Complement = false$), making the document classification problem a binary classification problem

$$\arg \max_{v \in \{true, false\}} P(Complement = v \mid d', \mathcal{D}). \quad (8.1)$$

Since it is non-trivial to get the necessary probability distributions, we solve the problem of Formula (8.1) by looking at SCDs, defining complementarity in terms of SCDs and a complement by using the notion of complementary SCDs. Based on these definitions, we specify a solution approach for corpus extension.

8.4.2. Complementary Documents

To classify a document as a complement, i.e., providing complementary content, with respect to the documents in the corpus of an agent, we need a formal definition of complementarity, for which we use the SCDs that are available in an SPO format and a taxonomy for interrelating entities occurring in them. As such, we transform the problem given in Formula (8.1) by defining a complement as a document with a complementarity value that exceeds a certain threshold. Focussing on SCDs in the *spo* format also has the advantage that we can automatically extract relational structures using available NE extraction methods such as OpenIE [AJPM15] to generate SCDs for documents. We can even use a lexical database of semantic relations and use hierarchies to interrelate those entities.

Before defining complementary of SCDs and documents, let us consider an example of a new document containing complementary content to the content of documents in a corpus. We pick up the example again in the course of this chapter.

Example 8.1. Assume that an agent is working with an individual collection of documents in corpus \mathcal{D} . The documents contain text about competitions at the Olympic Games 2021 in Tokyo. Thus, vocabulary $\mathcal{V}_{\mathcal{D}}$ is mainly characterized by words in the context of sports. The vocabulary of a new document d' giving a description about the occurrence of infection of SARS-CoV-2 in Tokyo is different

from $\mathcal{V}_{\mathcal{D}}$. In the context of similarity, d' would probably be classified as unrelated since the vocabularies $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{d'}$ might be very different. However, the content of d' might be complementary to the content of some documents in \mathcal{D} and thus, might support an agent to interpret content from documents in \mathcal{D} more suitably.

Definition 8.2 (Complementary SCDs). Given two documents d, d' and a taxonomy ξ , an SCD $t_i \in g(d')$ is complementary to an SCD $t_j \in g(d)$ if the entities in t_i and t_j are different but the entities are instances of the same concept or the predicates between the entities share a common meaning in ξ . Formally, the following seven types of complementarity between SCDs t_i and t_j exist (\uparrow refers to the concept in ξ that an entity belongs to):

- (1) s-complementary: $t_i = (s^\uparrow, p_i, o_i), t_j = (s^\uparrow, p_j, o_j)$,
- (2) p-complementary: $t_i = (s_i, p^\uparrow, o_i), t_j = (s_j, p^\uparrow, o_j)$,
- (3) o-complementary: $t_i = (s_i, p_i, o^\uparrow), t_j = (s_j, p_j, o^\uparrow)$,
- (4) sp-complementary: $t_i = (s^\uparrow, p^\uparrow, o_i), t_j = (s^\uparrow, p^\uparrow, o_j)$,
- (5) so-complementary: $t_i = (s^\uparrow, p_i, o^\uparrow), t_j = (s^\uparrow, p_j, o^\uparrow)$,
- (6) op-complementary: $t_i = (s_i, p^\uparrow, o^\uparrow), t_j = (s_j, p^\uparrow, o^\uparrow)$, and
- (7) spo-complementary: $t_i = (s^\uparrow, p^\uparrow, o^\uparrow), t_j = (s^\uparrow, p^\uparrow, o^\uparrow)$.

Let \mathcal{X} refer to the set of the different complementarity types $\{s, p, o, sp, so, op, spo\}$. An indicator function $\mathfrak{C}_x(t_i, t_j)$, $x \in \mathcal{X}$, returns 1 if t_i and t_j fulfill the conditions mentioned above for x -complementarity and otherwise 0, including when t_i or t_j is not in SPO format.

Generally, it might be possible to adapt the return value of the indicator function to include uncertainty by returning a value from $[0, 1]$. Next, we give an example on complementary SCDs.

Example 8.3 (Complementary SCDs). Assume that document d is in the agent's corpus and the agent is faced with a new document d' . Additionally, both documents are associated with SCDs yielding $g(d) = \{t_2, t_4\}$ and $g(d') = \{t_1, t_3\}$ where:

- $t_1 = (\textit{Olympic Games 2021}, in, Tokyo)$,
- $t_2 = (\textit{SARS-CoV-2}, spreading in, Tokyo)$,
- $t_3 = (\textit{UEFA Euro 2020}, in, Europe)$, and
- $t_4 = (\textit{Covid-19}, spreading in, London)$

8. Complementarity as an Inter-SCD Relation

Given the following taxonomy, where solid lines represent the hierarchy between classes and dashed lines represent instances of classes,



the indicator function $\mathfrak{C}_o(t_i, t_j)$ returns 1 for $i = 1$ and $j = 4$ since both `london` and `tokyo` are instances of class `city`. Additionally, the indicator function returns 1 for $i = 3$ and $j = 4$ since `london` is a `city` and `city` is a subclass of `continent`, to which `europe` belongs, too. Thus, t_1 and t_4 as well as t_3 and t_4 are o-complementary.

The different types of complementarity form a lattice as depicted in Figure 8.1, with the first three types, seen from the bottom, composing the lowest level, the next three types following on the next higher level, and the `spo`-type constituting the top entry. Up the lattice, the SCDs share more and more entities of the same concept, with the top entry requiring that the three positions are filled with different instances of the same concept, i.e., what falls under complementarity of higher levels also falls under complementarity of lower levels. This is different to Definition 1 of [BKB22], which requires all entities share the same concept or be *identical*. Definition 8.2 requires the entities to share the same concept or be *different*, thereby, further moving away from similarity to difference. A complementary SCD is now understood as a different SCD only sharing one or multiple identical concepts and not an identical SCD allowed to share one or multiple different concepts. Thus, the deviation of related and complementary SCDs will be much larger and the word vectors in the windows associated with the two types of SCDs will be more distinct. Next, we define complementary documents based on Definition 8.2.

Definition 8.4 (Complement). The complementarity value $\mathfrak{c}(d', d)$ between documents d' and d is given by

$$\mathfrak{c}(d', d) = \sum_{t_i \in g(d')} \sum_{t_j \in g(d)} \sum_{x \in \mathcal{X}} w_x \mathfrak{C}_x(t_i, t_j), \quad (8.2)$$

with $w_x \in [0, 1]$ a weight assigned to each complementarity type and $\sum_{x \in \mathcal{X}} w_x = 1$. Given a threshold θ_d , d' is complementary to d and thus called a *complement* if

$$\mathfrak{c}(d', d) > \theta_d. \quad (8.3)$$

Given the complementarity lattice, non-zero weights are only reasonable for types that do not subsume another. E.g., given spo-complementarity, w_x should be set to zero for all other complementarity types, i.e., $\forall x \in \mathcal{X}, x \neq \text{spo}$, as these types x would subsume spo. For the lowest level, $w_x = 0$ for all $x \in \{\text{sp}, \text{so}, \text{op}, \text{spo}\}$ while w_s , w_p , and w_o can be chosen freely as long as they add up to 1. Another possibility would be to have non-zero weights adding up to 1 for, e.g., sp and o, as they cover different positions in the triples and lie on different paths in the grid, with the remaining x set to 0. The threshold θ_d depends on a given corpus and adds subjectivity. θ_d can be adjusted according to the agent's need for new documents. With a need for more documents, an agent may choose a low threshold in combination with the broadest senses of complementarity, s, p and o. Aiming for adding only a few documents in the more immediate context, a high threshold and spo-complementarity might be a fitting choice. In Example 8.3, $\mathfrak{c}(d', d) = 3$ with $w_o = 1$ ($\mathfrak{C}_o(t_2, t_1) = 0$). In all other cases, $\mathfrak{c}(d', d) = 0$ as $\forall x \neq o : \mathfrak{C}_x(t_i, t_j) = 0$ with the present taxonomy.

We use the decision criterion in Formula (8.3) to solve the document classification problem of Formula (8.1). Within the framework of Kuhr et al.'s document classification problem, we could focus computing Formula (8.3) for those documents that are otherwise classified as unrelated, making a distinction between complement and unrelated. How well this definition works for distinguishing unrelated and complementary documents is showcased during the evaluation in Section 8.6. But before, we present how to use the definitions of complementarity for the task of corpus extension and briefly discuss what else can be done having complementarity available.

8.4.3. Corpus Extension with Complements

Corpus extension as a task so far has used similarity values, specifically the sequence of MPS²CD similarity values over a document in order to classify an unknown document as either of the document types of *sim*, *ext*, *rev*, and *unrel*, and then decide

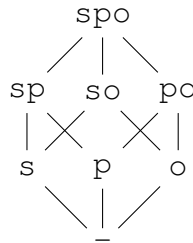


Figure 8.1.: The complementarity types of Definition 8.2 in a lattice.

8. Complementarity as an Inter-SCD Relation

Algorithm 10 Corpus Extension with Complements

```

1: function EXTENDCOMPLEMENT( $\mathcal{D}$ ,  $d'$ ,  $\theta_{\mathcal{D}}$ ,  $\{w_x\}_{x \in \mathcal{X}}$ )
2:   Input: Corpus  $\mathcal{D}$ ; New document  $d'$ ; Threshold  $\theta_{\mathcal{D}}$ ; Weights  $\{w_x\}_{x \in \mathcal{X}}$ 
3:   Output: true (complement/extend) or false (no complement/extend not)
4:   if  $g(d') = \emptyset$  then
5:     Add SCDs (i.e., spo-triples) to  $d'$  using OpenIE
6:    $c \leftarrow 0$ 
7:   for each  $t_i \in g(d')$  do
8:     for each  $d \in \mathcal{D}$  do
9:       for each  $t_j \in g(d)$  do
10:        for each  $x \in \mathcal{X}$  do
11:           $c \leftarrow c + w_x \mathfrak{C}_x(t_i, t_j)$ 
12:   if  $c > \theta_{\mathcal{D}}$  then
13:     return true
14:   else
15:     return false

```

its inclusion based on this outcome. Similar and unrelated documents were ignored whereas extensions and revisions were added or exchanged with the originals. An agent performing corpus extension with complementary documents has to answer the same question about possibly including an unknown document. However, now the agent aims to extend its corpus with complements. To perform the task, the agent applies the definitions above for reaching a decision.

Algorithm 10 shows an outline of the workflow the agent follows when presented with an unknown document d' for possible inclusion into its corpus \mathcal{D} on the condition that d' is a complement in \mathcal{D} . The algorithm uses a corpus-specific threshold $\theta_{\mathcal{D}}$, which fulfils the same role as the threshold θ_d in Formula (8.3) but factors in that it applies to the whole corpus and not a single document. The first if-condition asks whether d' already contains SCDs. If not, the agent uses OpenIE to extract *spo*-triples from the text of d' . Then follows a for-loop that accumulates the complementarity values for each SCD t_i associated with d' over all documents in \mathcal{D} . Afterwards, the agent tests the accumulated value against $\theta_{\mathcal{D}}$ to return *true* if it considers d' a complement based on Definition 8.4, and *false* otherwise.

8.4.4. Discussion

The following paragraphs discuss complements as part of the general classification problem for the task of document retrieval as well as for augmenting user output by returning positions of interest.

Complements as a Document Type While we provide a more general approach in the upcoming section, a direct way to introduce complements as category *compl* into the corpus extension by Kuhr et al. [KBBM19, KBBM20] is the following: Their classification problem is defined given a sequence of MPS²CD similarity values \mathcal{W} computed by Algorithm 2 for an unknown document d' and a corpus \mathcal{D} :

$$\arg \max_{y \in \mathcal{Y}} P(\text{Type} = y \mid \mathcal{W}), \quad (8.4)$$

with $\mathcal{Y} = \{sim, ext, rev, unrel\}$. Generalizing and merging Formulas (8.4) and (8.1), we could formulate the classification problem as follows:

$$\arg \max_{y \in \mathcal{Y}} P(\text{Type} = y \mid d', \mathcal{D}), \quad (8.5)$$

with $\mathcal{Y} = \{sim, ext, rev, unrel, compl\}$. In Formula (8.5), an unknown document and the corpus are given. A reasonable workflow to classify an unknown document would then be to use the document type detection algorithm in [KBBM19, KBBM20] and then apply Algorithm 10 to the unknown document if the previous classification returns *unrel*.

Document Retrieval For document retrieval in the context of complementarity, i.e., complement retrieval, a user could provide a document d' for which they want k complementary documents returned from the corpus \mathcal{D} available to the agent. The agent would then calculate complementarity values for d' compared to each document $d_i \in \mathcal{D}$, i.e., $c(d', d_i)$ following Definition 8.4, and return the top- k documents, i.e., those k documents with the highest complementarity values. In contrast to Algorithm 10, the agent would not sum up the complementarity values but rather store the current top- k documents with their complementarity value and test whether the next document d_{i+1} has a higher value than the lowest value currently stored and replace that document if true.

Augmenting Enrichment: Positions of Interest In general, it is difficult to understand the reason a new document is classified as a complementary document by looking at the content of the document. The only thing we know for a document being classified as complementary is that some entities from a new document share a class with entities from documents in the corpus. Thus, one might highlight complementary SCDs such that it is possible to identify the positions in a text that are relevant for Algorithm 10 classifying a document as a complementary document. We denote those positions as *positions of interest*.

Identifying complementarity in documents is beneficial for IR agents in several ways, e.g., for deciding corpus extension, retrieving documents in response to queries, and identifying relevant positions in the documents.

With the definition of complementarity in place and a solution approach specified with Algorithm 10, we are able to detect complementary documents. However, the classification process is not straightforward, as first the documents are classified as one of $\{sim, ext, rev, unrel\}$. Then, if a document is classified as *unrel*, a second classification between $\{unrel, compl\}$ is performed. In the next section, we describe how to integrate complementarity in the SCD matrix, allowing for directly classifying documents as one of $\{sim, ext, rev, unrel, compl\}$.

8.5. Document Classification with Complementarity and Similarity

The SCD matrix generally only contains related SCDs, thus we call it rSCD matrix. Kuhr et al. classify a new document d' using similarity by the four document types $\{sim, ext, rev, unrel\}$. To support complementarity, we propose a combined SCD (cSCD) matrix, containing related and complementary SCDs in one matrix. The cSCD matrix allows an agent to classify a new document d' by five document types $\{sim, ext, rev, unrel, compl\}$ in one classification process. First, we present the cSCD matrix and an algorithm to build it, followed by a filtering technique removing noisy SCDs from a cSCD matrix using the definition of complementarity, and finally, a straightforward classification process.

8.5.1. Combined SCD Matrix

The cSCD matrix combines two corpora: \mathcal{D}_r contains the related documents from the agent's corpus, the same corpus Kuhr et al. train their matrix on. The cSCD matrix is additionally trained on \mathcal{D}_c containing complementary documents to the agent's corpus. Corpus \mathcal{D}_c can be either formed by using Algorithm 10 or by using expert's knowledge, i.e., by manually forming a corpus of complementary documents. Analogously to the general notations, each corpus has a set of SCDs $t_j^r \in g(\mathcal{D}_r)$ and $t_j^c \in g(\mathcal{D}_c)$. The vocabularies of both corpora are joined, i.e., $\mathcal{V} = \mathcal{V}_{\mathcal{D}_r} \cup \mathcal{V}_{\mathcal{D}_c}$ and $L = |\mathcal{V}|$. The cSCD matrix $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$ is a more specific form of $\delta(\mathcal{D})$ described in Section 3.1.

The first K_r rows belong to the SCDs $g(\mathcal{D}_r)$ and the last K_c rows belong to $g(\mathcal{D}_c)$. Each row in the matrix forms an SCD-word distribution vector, again using counts of words. The vector entry itself represents a probability value describing how likely it is that a word occurs in an SCD window surrounding the position associated with the SCD, yielding an SCD-word probability distribution for each SCD associated with documents in \mathcal{D}_r or \mathcal{D}_c , respectively.

$$\delta_c(\mathcal{D}_r, \mathcal{D}_c) = \begin{matrix} & w_1 & w_2 & w_3 & \cdots & w_L \\ \begin{matrix} t_1^r \\ \vdots \\ t_{K_r}^r \\ t_1^c \\ \vdots \\ t_{K_c}^c \end{matrix} & \left(\begin{matrix} v_{1,1} & v_{1,2} & v_{1,3} & \cdots & v_{1,L} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{K_r,1} & v_{K_r,2} & v_{K_r,3} & \cdots & v_{K_r,L} \\ v_{K_r+1,1} & v_{K_r+1,2} & v_{K_r+1,3} & \cdots & v_{K_r+1,L} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{K_r+K_c,1} & v_{K_r+K_c,2} & v_{K_r+K_c,3} & \cdots & v_{K_r+K_c,L} \end{matrix} \right) \end{matrix}$$

Furthermore, associated with a cSCD matrix is a set of complementarity relations between the SCDs of the matrix:

$$\mathcal{C}_x = \{(\{t_i^r, t_j^c\}, \mathfrak{C}_x(t_i^r, t_j^c)) \mid t_i^r \in g(\mathcal{D}_r), t_j^c \in g(\mathcal{D}_c)\}$$

where $x \in \mathcal{X}$ refers to a complementarity type and \mathfrak{C}_x returns a continuous value from $[0, 1]$. Note that the complementarity value between two SCDs is symmetric, i.e., $\mathfrak{C}_x(t_i, t_j) = \mathfrak{C}_x(t_j, t_i)$ for any SCDs t_i, t_j . For each SCD t_i^r , the complementarity set \mathcal{C}_x represents the corresponding complementary SCDs, e.g., t_j^c together with the value of complementarity $\mathfrak{C}_x(t_i^r, t_j^c)$. Thus, given an SCD linked to a document, it is possible to retrieve the complementary SCDs and linked complementary documents. The set \mathcal{C}_x can also be represented by a $K_r \times K_c$ matrix with the complementarity values stored in the cells.

Building Combined SCD Matrices

The Supervised Estimator of cSCD Matrices (SEcM), Algorithm 11, creates a cSCD matrix and the associated set \mathcal{C}_x based on two corpora \mathcal{D}_r and \mathcal{D}_c . SEcM iterates over all SCDs of both corpora and updates for each sentence the word count vector based on the influence value I . Afterwards, the relations between the SCDs in the matrix are calculated applying Definition 8.2 and the complementarity values are stored in \mathcal{C}_x .

If the corpora are not associated with SCDs, first USEM can be used on each corpus separately. After running USEM, OpenIE can be used to add *spo*-triples to each SCD. Finally, the cSCD matrix can be estimated by SEcM.

8. Complementarity as an Inter-SCD Relation

Algorithm 11 Supervised Estimator of cSCD Matrices $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$

```

1: function SECM( $\mathcal{D}_r, \mathcal{D}_c, x$ )
2:   Input: Corpora  $\mathcal{D}_r, \mathcal{D}_c$ ; complementarity type  $x$ 
3:   Output: cSCD matrix  $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$ ; complementarity relations  $\mathcal{C}_x$ 
4:   Initialize a  $(K_r + K_c) \times L$  matrix  $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$  with zeros
5:   for each  $d \in \mathcal{D}_r \cup \mathcal{D}_c$  do ▷ Form word distributions
6:     for each  $t \in g(d)$  do
7:       for referenced sentence  $s \in t$  do
8:         for each word  $w_i \in s$  do
9:            $\delta_c(\mathcal{D}_r, \mathcal{D}_c)[t][w_i] += I(w_i, s)$ 
10:  Normalize  $\delta_c(\mathcal{D}_r, \mathcal{D}_c)[t]$  ▷ Skipped for matrix to contain counts
11:  Initialize  $\mathcal{C}_x \leftarrow \emptyset$ 
12:  for each  $t^r \in g(\mathcal{D}_r)$  do ▷ Extract complementarity relations
13:    for each  $t^c \in g(\mathcal{D}_c)$  do
14:       $\mathcal{C}_x \leftarrow \mathcal{C}_x \cup \{(\{t^r, t^c\}, \mathfrak{C}_x(t^r, t^c))\}$  ▷ Indicator function Definition 8.2
15:  return  $\delta_c(\mathcal{D}_r, \mathcal{D}_c), \mathcal{C}_x$ 

```

Filtering Combined SCD Matrices

A cSCD matrix formed by SEcM contains a vector (row) for each SCD from both corpora. However, complementary documents also contain some false-complementary SCDs, i.e., SCDs for general sentences which can occur in documents of any context. Such false-complementary SCDs are similar to noisy data because the false-complementary SCDs are considered as complementary when using the cSCD matrix, even though they are not. Depending on the use-case, false-complementary SCDs might provide useful data but in our scenario they add noise to the results. Therefore, we introduce the filtered cSCD (cSCD^f) matrix, in which the false-complementary SCDs are removed from the matrix. In the evaluation, we compare the performance of the cSCD and cSCD^f matrix for classification.

During filtering, all SCDs in $g(\mathcal{D}_r)$ are kept. SCDs in $g(\mathcal{D}_c)$, which are not complementary by our definition of complementarity, are removed. We use a threshold θ_δ to decide if an SCD is complementary, which depends on the corpora, complementarity type, and cSCD matrix used. Algorithm 12 iterates over all SCDs $t^c \in g(\mathcal{D}_c)$ and extracts the highest complementarity value of each t^c to any SCD $t^r \in g(\mathcal{D}_r)$. If the highest value is smaller than θ_δ , the SCD t^c is considered to be false-complementary and the SCD-word distribution $\delta_c(\mathcal{D}_r, \mathcal{D}_c)[t^c]$ is not included in the cSCD^f matrix.

To decide about document extension, an agent faced with an unknown document d' has to identify the document type $y \in \mathcal{Y} = \{sim, ext, rev, unrel, compl\}$ of d' , with \mathcal{Y} now containing *compl* as another type compared to previous settings. For

Algorithm 12 Filter cSCD matrix to get cSCD^f matrix

```

1: function FILTERCOMBINEDMATRIX( $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$ ,  $\mathcal{C}_x$ ,  $\theta_\delta$ )
2:   Input: cSCD matrix  $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$ ; complementarity relations  $\mathcal{C}_x$ ; threshold  $\theta_\delta$ 
3:   Output: cSCDf matrix  $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$ 
4:   for each  $t^c \in g(\mathcal{D}_c)$  do
5:      $best \leftarrow 0$ 
6:     for each  $t^r \in g(\mathcal{D}_r)$  do
7:        $value \leftarrow \mathcal{C}_x(t^c, t^r)$   $\triangleright$  Retrieve complementarity value from set  $\mathcal{C}_x$ 
8:        $best \leftarrow \max\{value, best\}$ 
9:     if  $best < \theta_\delta$  then
10:      Delete row  $\delta_c(\mathcal{D}_r, \mathcal{D}_c)[t^c]$ 
11:   return  $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$ 

```

document type identification, \mathcal{C}_x is no longer needed after filtering. We have to adapt existing procedures to comply with complementarity, which also includes the switch from rSCD to cSCD or cSCD^f matrices. As the existing procedure is based on MPS²CDs and their similarity values over the text of d' , we first consider how to estimate MPS²CDs for a d' given a cSCD or cSCD^f matrix of a corpus $\mathcal{D}_r \cup \mathcal{D}_c$.

8.5.2. Estimate Most Probably Suited SCDs in cSCD Matrices

To estimate MPS²CDs, the agent generates a word count vector from the words of each sentence of the new document. The agent compares the word count vector of each sentence with the word count vector of each SCD associated with documents in the corpus. The SCD where the word count vector has the smallest distance (highest cosine similarity) to the word count vector of the sentence is associated with the sentence. We refer to this associated SCD as the combined MPS²CD (cMPS²CD). A cMPS²CD may originate from the related or complementary corpus. Thus, a new document associated with mostly complementary cMPS²CDs might be by a complementary document, while a new document associated with mostly related cMPS²CDs is more likely a similar or revised document.

Algorithm 13 outlines the procedure of estimating cMPS²CDs, which not only returns the cMPS²CDs but also the cosine similarity values of each cMPS²CD and hence of each sentence in d' . We call the sequence of cosine similarity values for a document cMPS²CD similarity sequence or in short cMPS²CD similarities. Each sentence is associated with a cMPS²CD similarity value.

During the estimation of cMPS²CDs, we map the similarity values of related SCDs to the interval $[0, 1]$ as before, with 0 representing *unrelated* SCDs. In contrast,

Algorithm 13 Estimating MPS²CDs using cSCD and cSCD^f matrices

```

1: function COMBINEDMPS2CD( $d'$ ,  $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$ )
2:   Input: Document  $d'$ ; cSCD or cSCDf matrix  $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$ 
3:   Output: SCDs  $g(d')$  with similarity values  $\mathcal{W}$ 
4:    $\mathcal{W} \leftarrow \emptyset$ 
5:    $g(d') \leftarrow \emptyset$ 
6:   for each sentence  $s_i^{d'} \in d'$  do
7:      $\delta(s_i^{d'}) \leftarrow$  new zero-vector of length  $L$ 
8:     for each word  $w \in s_i^{d'}$  do
9:        $\delta(s_i^{d'})[w] += I(w, s_i^{d'})$ 

10:    $t' \leftarrow \arg \max_{t \in g(\mathcal{D}_r) \cup g(\mathcal{D}_c)} \frac{\delta_c(\mathcal{D}_r, \mathcal{D}_c)[t] \cdot \delta(s_i^{d'})}{\|\delta_c(\mathcal{D}_r, \mathcal{D}_c)[t]\|_2 \cdot \|\delta(s_i^{d'})\|_2}$ 
11:    $sim \leftarrow \max_{t \in g(\mathcal{D}_r) \cup g(\mathcal{D}_c)} \frac{\delta_c(\mathcal{D}_r, \mathcal{D}_c)[t] \cdot \delta(s_i^{d'})}{\|\delta_c(\mathcal{D}_r, \mathcal{D}_c)[t]\|_2 \cdot \|\delta(s_i^{d'})\|_2}$ 
12:   if  $t \in g(\mathcal{D}_c)$  then ▷ Negative value for complementary SCDs
13:      $sim \leftarrow sim \cdot -1$ 

14:    $g(d') \leftarrow g(d') \cup t'$  ▷ Also add  $s_i^{d'}$  as referenced sentence to  $t'$ 
15:    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(t', sim)\}$ 
16:   return  $g(d'), \mathcal{W}$ 

```

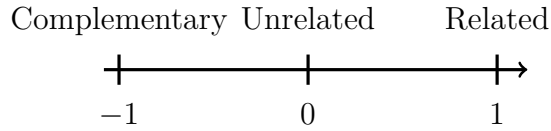


Figure 8.2.: Range of the MPS²CD similarity values along with their interpretations.

complementarity is represented by the interval $[-1, 0]$ left of 0, with 0 representing *not complementary*, i.e., unrelated in terms of complementary SCDs, again. Thus, we overall get the interval $[-1, 1]$ for the cMPS²CD similarity values. The complementarity of sentences is now expressed by numbers in $[-1, 1]$, where -1 stands for complementary and 1 for related, with 0 the point of intersection for *unrelated* SCDs. Figure 8.2 illustrates the interval of the cMPS²CD similarity values and Algorithm 13 can be imagined as mapping each sentence to the interval.

8.5.3. Classifying Documents

Using the previously described techniques, a cSCD and cSCD^f matrix can be trained and a sequence of cMPS²CDs similarities can be estimated for a new document. Similarly to Kuhr et al., we apply an ensemble of HMM to analyze the cMPS²CD similarities.

The cMPS²CD similarities contain for each sentence a similarity value, which aligns each sentence between complementary and related whereas unrelated is in the middle. Given this sequence describing a new document, we need to classify the document by five document types. For each type, we assume the following behavior of the sequences:

- unrel* An unrelated document results in a sequence with mostly small values around 0. The word count vectors of the sentences neither match the vectors of the related nor the complementary SCDs in the matrix.
- compl* A complementary document results in a sequence with mostly high negative values close to -1 . The word vectors of the complementary SCDs in the matrix are more similar to the vectors created on the sentences.
- sim* While a complementary document results in negative values, a similar document results in high positive values close to 1.
- ext* An extended document shows two sectors: The sequence starts with high positive values and ends with smaller or even negative values, if extended with complementary content.
- rev* In a revised document sentences have been replaced by complementary or unrelated content while the remaining sentences remain related. Thus, the sequence contains high positive, high negative, and even small values.

According to our assumptions about the sequences, we define a suitable HMM.

Definition 8.5 (Hidden Markov model). An HMM λ for classifying documents is a tuple $(\Omega, \Delta, A, B, \pi)$ consisting of

- (hidden) states $\Omega = \{\omega_1, \omega_2, \omega_3\}$, with state ω_1 representing complementary, ω_2 unrelated, and ω_3 related sentences,
- an observation alphabet $\Delta = \{o_{-m}, \dots, o_0, \dots, o_m\}$, where each o_i represents a range of MPS²CD similarity values; the observation alphabet is generated by discretizing cMPS²CD similarity values,
- a transition probability matrix A representing the probability of all possible state transitions $a_{i,j}, i, j \in \{1, 2, 3\}$ between the three states $\omega_1, \omega_2, \omega_3 \in \Omega$, which implies moving forward in time from time step t to $t + 1$,
- an emission probability matrix B representing the probability of emitting a symbol from observation alphabet Δ for each possible state in Ω , and
- an initial state distribution vector $\pi = \pi_0$.

With $\sum_{j=1}^3 a_{i,j} = 1$ for each $\omega_i \in \Omega$ summing over Ω , the entries of A between states $\omega_i, \omega_j \in \Omega$, represent the following conditional probability:

$$a_{i,j} = P(\omega_j | \omega_i).$$

With $\sum_{k=-m}^m b_j(o_k) = 1$ for each $\omega_j \in \Omega$ summing over Δ , the entries of B represent the following conditional probability:

$$b_j(o_k) = P(o_k | \omega_j).$$

The semantics of λ is given by unrolling λ for a given number of time steps and building a full joint distribution.

We compose an HMM of three hidden states because we assume each sentence represented may be related, unrelated, or complementary. The discrete observation alphabet Δ requires discretizing the sequences of cMPS²CD similarities. A discretization function $f : [-1, 1] \mapsto \Delta$ maps each cMPS²CD similarity value sim to one of the symbols in Δ based on m thresholds th_1, \dots, th_m :

$$f(sim) = \begin{cases} o_{-m} & -1 \leq sim < -th_m \\ \vdots & \\ o_0 & -th_1 \leq sim < th_1 \\ \vdots & \\ o_m & th_m \leq sim \leq 1 \end{cases}$$

Algorithm 14 Classification using an ensemble of HMMs and MPS²CD similarities

```

1: function CLASSIFYDOCUMENT( $d'$ ,  $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$ ,  $\mathcal{H}$ )
2:   Input: Document  $d'$ ; cSCD or cSCDf matrix  $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$ ; HMMs  $\mathcal{H}$ 
3:   Output: Type  $y \in \mathcal{Y}$  of document  $d'$ 
4:    $y \leftarrow nil$ ,  $p \leftarrow 0$ 
5:    $\mathcal{W} \leftarrow \text{COMBINEDMPS}^2\text{CD}(d', \delta_c(\mathcal{D}_r, \mathcal{D}_c))$ 
6:    $O \leftarrow \text{DISCRETIZE}(\mathcal{W})$ 
7:   for each HMM  $\lambda_y \in \mathcal{H}$  do
8:      $v \leftarrow \text{VITERBIPROBABILITY}(\lambda_y, O)$ 
9:     if  $v > p$  then
10:        $p \leftarrow v$ 
11:        $y \leftarrow \lambda_y$ 
12:   return  $y$ 

```

In general, the transition probability matrix A and the emission probability matrix B are unknown and have to be learned, e.g., using the Baum-Welch algorithm [Bau72]. Using a set of documents with known document type

$$y \in \mathcal{Y} = \{sim, ext, rev, unrel, compl\}$$

we calculate the MPS²CD similarities, discretize them, and train an HMM for each document type. The resulting ensemble of five HMMs

$$\mathcal{H} = \{\lambda_y | y \in \mathcal{Y}\}$$

is used by Algorithm 14 to classify a new document.

Algorithm 14 estimates and discretizes the MPS²CD similarities for a new document and runs the Viterbi algorithm [Vit67] for each HMM in \mathcal{H} . The Viterbi algorithm calculates the most probable sequence of hidden states on each HMM for the given sequence of observation symbols. Thus, the Viterbi algorithm gives for each HMM the overall probability that this HMM creates the observed sequence of similarities. Finally, a document is classified as the document type for which the document type's HMM yielded the highest probability.

In summary, this section presents an improved version of the document classification by Kuhr et al. [KBBM20] to also recognize complementarity. In the next section, we provide an evaluation comparing the classification performance of documents using the rSCD, cSCD, and cSCD^f matrix.

8.6. Evaluation

In this section, we present an evaluation illustrating the potential of the definition of complementarity and the classification approach using cSCD and cSCD^f matrices. We demonstrate that document classification using only an rSCD matrix is not able to detect complementary documents well. We show that the cSCD and cSCD^f matrices perform significantly better on the problem. Before we look at the results, we describe the corpus and workflow used in the evaluation.

8.6.1. Corpus

In this evaluation, we use articles from the English Wikipedia as documents in a corpus. All documents in the corpus contain text about car manufacturers². Thus, documents about car manufacturers are related documents. We manually create document extensions by concatenating related and unrelated documents. To create a revised version of a document, we replace 40% of the sentences in related documents with sentences from unrelated documents. The class of unrelated documents contains the following 16 Wikipedia articles: *Apple Inc.*, *Apple*, *iPhone*, *Microsoft Windows*, *Google*, *Donald Trump*, *Atlantic Ocean*, *Angela Merkel*, *Baltic Sea*, *SpaceX*, *Lawyer*, *Titanic*, *Management*, *President (government title)*, *Mountain*, and *Snow*. Wikipedia articles about the cities where each of the car manufacturers' headquarters are located act as complementary documents. For example, the document *Toyota City, Aichi, Japan* is complementary to *Toyota Motor*.

Generally, the context of the corpus we are interested in can be described by *cars and their manufacturers*. Unrelated documents like *Apple Inc.* do not represent the manufacturing of cars and a profession like *Lawyer* neither represents cars nor manufacturing. We argue that complementary documents used in the evaluation fulfill our definition of complements, as the production of cars influences the city where the manufacturer is located, e.g., employees working at the manufacturer will reside in the city, the manufacturer pays taxes, and geographical conditions or historical circumstances of the city may originate from the manufacturer. However, some of the unrelated documents might be also a bit complementary, e.g., *Apple Inc.* contains a short paragraph about an autonomous car. In contrast, the document about the fruit *Apple* contains no content about cars or manufacturers. In this manner, it is important to notice that our definition of complementarity is universal and is not dependent or trained on a specific corpus. However, the cSCD and cSCD^f matrices are computed to fit the selected corpus.

²<https://w.wiki/4FUS>

8.6.2. Workflow and Implementation

All algorithms are implemented using Python. We apply OpenIE [AJPM15] to extract SCDs in the *spo* format from each window over the word sequences from the articles. Additionally, the WordNet [Mil95] interface, provided by the Natural Language Toolkit³, is used to detect if different entities share the same concept or a common meaning in the sense of Definition 8.2.

The implementation is optimized for speed and runs on multiple processor cores. It uses the libraries Gensim⁴, NumPy⁵, SciPy⁶ and Pomegranate⁷. We run all experiments in a Docker container on a machine featuring 8 Intel 6248 cores at 2.50GHz (up to 3.90GHz) and 16GB RAM.

Before forming the SCD matrices using Algorithm 11, all documents are preprocessed by (i) removing punctuation, (ii) lowercasing all characters, (iii) stemming words, (iv) tokenizing the result, and (v) eliminating tokens from a stop-word list containing 179 words. OpenIE and WordNet’s morphological processing tool Morpho use their own default preprocessing.

Each technique to test has its own workflow: (i) *similarity-based* serves as baseline and equals Kuhr et al. [KBBM20], (ii) *complementarity-based* directly uses the definition of complementary documents, and (iii) *document classification* describes how we apply the cSCD and cSCD^f matrix.

Similarity-based

Training and using a similarity-based rSCD matrix works similar to Algorithms 11, 13, and 14 with three differences, (i) no set \mathcal{C} is generated, (ii) the set \mathcal{D}_c is empty, and (iii) the HMMs have only two states (related and unrelated).

We interpret the probability of the most likely sequence of an HMM for a sequence of observations as sequence similarity. Using this similarity, the new documents are classified, e.g., by taking the most probable HMM’s document type or using a threshold on the similarity value.

³<https://www.nltk.org/>

⁴<https://radimrehurek.com/gensim/>

⁵<https://numpy.org/>

⁶<https://www.scipy.org/>

⁷<https://pomegranate.readthedocs.org/>

Complementarity-based

First, an rSCD matrix is used to classify truly unrelated documents and complementary documents into a single class *unrel*. Definition 8.4 detects complementary documents and thus allows to separate truly unrelated documents from complementary documents. The implementation of Algorithm 10 considers each pair of SCDs, i.e., *spo*-triples, between $t_i^r \in g(d^r)$, $d^r \in \mathcal{D}_r$ and $t_j \in g(d')$. Due to the huge amount of pairs, the implementation randomly samples 100 pairs from each set and consider each of their combinations. Then, all complementarity types $x \in \{s, p, o, sp, so, op, spo\}$ are computed for each pair t_i^r, t_j , while returning continuous values from the indicator function \mathfrak{C}_x . For each item in the *spo*-triples of t_i^r, t_j , the tool Morphy extracts matching entities in WordNet. If there are multiple possible entities, the implementation considers all possible entities and uses the path similarity from WordNet to detect if the entities share the same concept or a common meaning. Entities with path similarities smaller than 0.1 are treated as different. Finally, the average or maximum across all path similarities is returned as complementarity value \mathfrak{C}_x . For example, if the object of t_i is represented by the entities e_1, e_2 and the object of t_j by the entities e'_1, e'_2 , the complementarity value max is given by $\max\{sim_{path}(e_1, e'_1), sim_{path}(e_1, e'_2), sim_{path}(e_2, e'_1), sim_{path}(e_2, e'_2)\}$. Our implementation normalizes the complementarity values after each sum of Definition 8.4 such that $\mathfrak{c}(d', d) \in [0, 1]$. Furthermore, it also calculates $\mathfrak{c}_x(d', d)$ only considering complementarity type x , i.e., $w_x = 1$ and $w_{x'} = 0 \ \forall x' \in \mathcal{X} \setminus x$.

Document Classification

To classify documents as one of the five document types, we apply Algorithms 11, 12, 13, and 14. We use the corpus about car manufacturers and train a cSCD and cSCD^f matrix on eight documents about car manufacturers and use eight cities as complementary documents during the training. For becoming the baseline, an rSCD matrix on the same eight documents about car manufacturers is trained. Additionally, we build corpora of eight documents for each of the document types, these five corpora are disjoint to the corpora used while training the matrices. We evaluate with four cycles of training HMMs and testing their classification performance. In each cycle the documents are randomly split into four documents for training and four documents for testing. Afterwards, the the average across the results of the four cycles is used. As described for the complementarity-based approach in the previous paragraph, the implementation uses the path similarity of WordNet but returns the average across all path similarities as the complementarity value \mathfrak{C}_x in the set of relations \mathcal{C}_x .

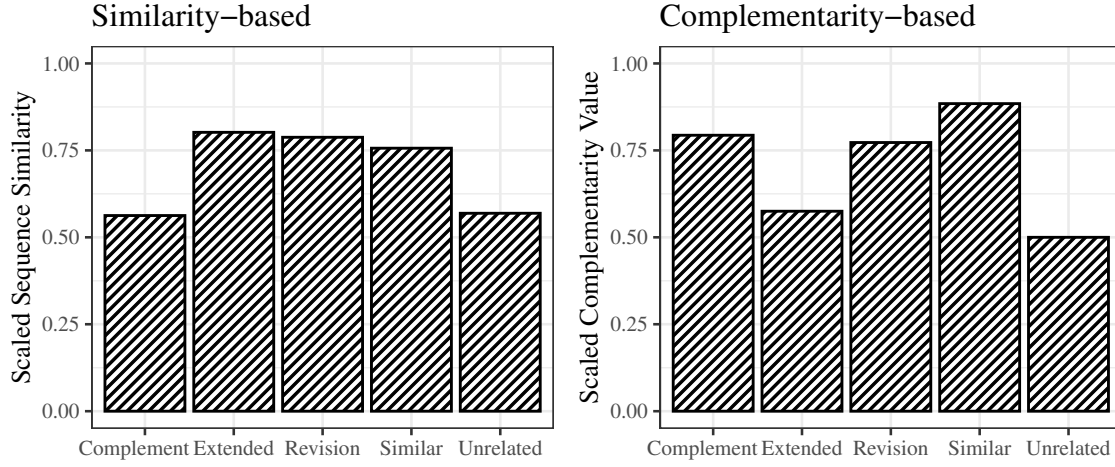


Figure 8.3.: Left: Average scaled similarity values per document type gained from the similarity-based approach using an rSCD matrix. Right: Average scaled complementarity values yielded by Definition 8.4.

8.6.3. Results

We present the result in three parts: First, we only compare the similarity and complementarity values of the differences document types. Second, we present the classification performance of documents using an rSCD, cSCD, and cSCD^f matrix. Finally, we illustrate which algorithms are needed by the techniques online and offline, including consequences on the runtime.

In Figure 8.3, the similarity values and complementarity values are scaled to the interval $[0, 1]$. In the left plot, the sequence similarities gained from the similarity-based approach are shown for all five document types. The similarity value of *compl* and *unrel* documents is nearly equal. Thus, it is not possible to detect complementary documents using the similarity-based approach presented in [KBBM20]. In the right plot, the complementarity values $\max \mathbf{c}_{op}(d', d)$ are shown for all five classes. Complementary documents have a much higher value than unrelated documents, therefore it is possible to separate complementary documents from unrelated documents using a threshold $\theta_{\mathcal{D}}$ in Algorithm 10. Interestingly, extended documents are nearly as complementary as unrelated documents and revisions are similar to complements using our definition of complementarity.

In Figure 8.4, we compare the performance of an rSCD matrix with a cSCD and cSCD^f matrix. For all five document types and all three matrices the accuracy is shown. In all cases, the cSCD^f matrix results in the best values, except for complementary documents. Complementary documents are classified best by the

8. Complementarity as an Inter-SCD Relation

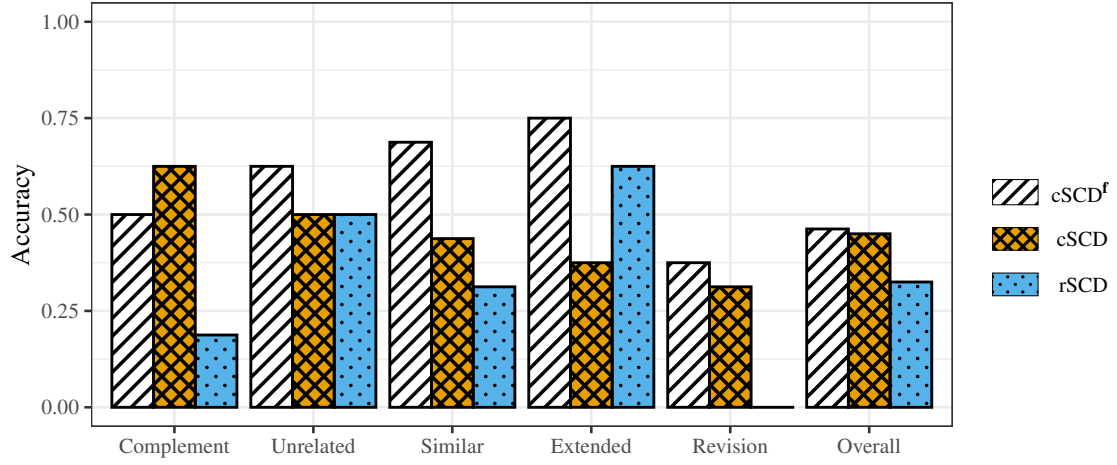


Figure 8.4.: Accuracy classifying each document type using a cSCD^f, cSCD, or rSCD matrix.

cSCD matrix. Presumably, the cSCD matrix contains more relevant information about complementary documents than the cSCD^f matrix after the filtering. Overall, the cSCD^f matrix performs best, the rSCD matrix worst and the cSCD in between.

In general, the accuracy values in this five-classes setting are lower than the results gained by Kuhr et al. with four classes. Adding a fifth type of documents increases the difficulty of the classification problem. Randomly choosing one of five types would result in an accuracy of 0.2 while we reach accuracies between around 0.4 and 0.75. Compared to Kuhr et al., we use sentences, i.e., a form of a tumbling window, instead of sliding windows over the text, which might also influence the accuracy as well. However, using the cSCD^f (and cSCD) matrix improves the accuracy values significantly in our evaluation, especially compared to the rSCD matrix, which is the basis of the approach by Kuhr et al.

In Figure 8.5, precision, recall and F1-Score are shown for each document type and matrix. Again, we notice that the cSCD matrix works best on the complement document type while the cSCD^f matrix works best overall. Especially, similar, unrelated, and complementary documents are classified well. Classifying extended and revised documents seems to be more difficult because they consist of related, unrelated, and maybe even complementary sentences. For an HMM, it is difficult to model the cMPS²CD similarities of a revised document, because the sequence may contain high positive, high negative, and even small values that do not follow any scheme. Working with sliding windows or larger amounts of documents would result in more data, which can be used to train the cSCD or cSCD^f matrices and might lead to overall better results.

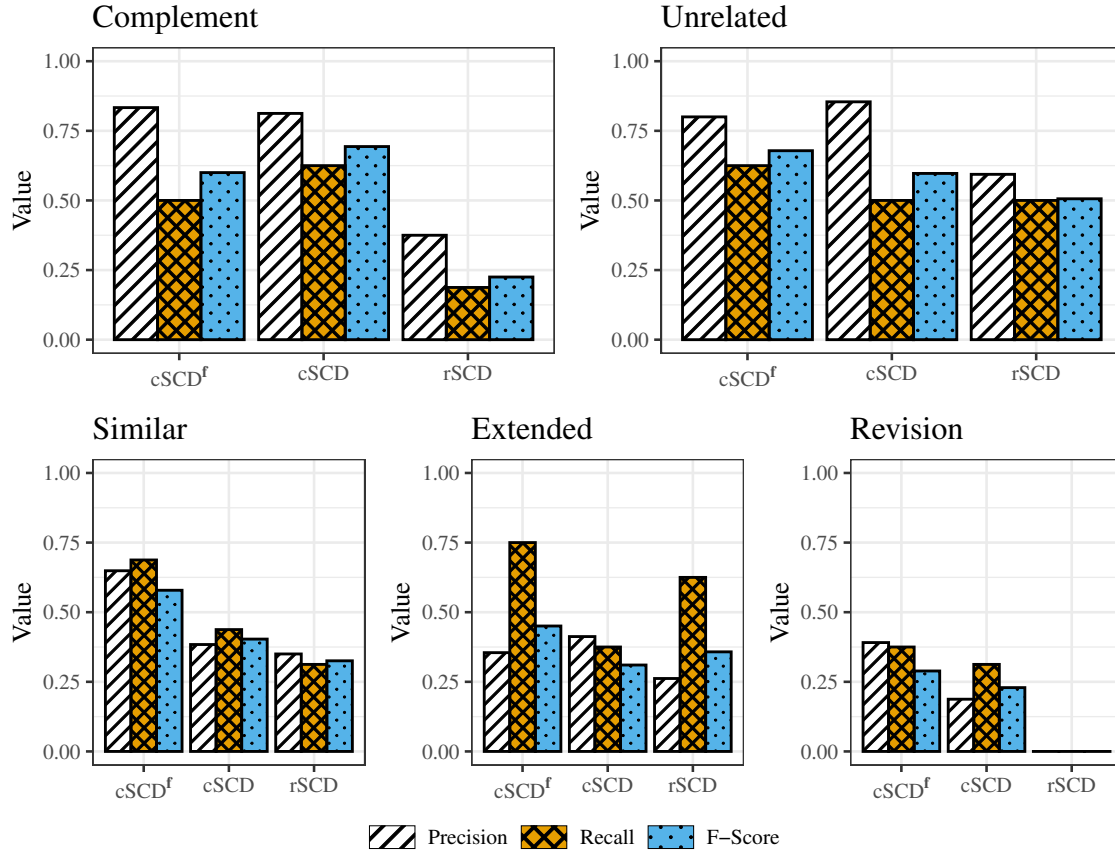


Figure 8.5.: Precision, recall, and F1-Score classifying each document type using a $cSCD^f$, $cSCD$, or $rSCD$ matrix.

Technique Algorithm	Definition 8.4 only		rSCD		cSCD		cSCD ^f	
	Offline	Online	Offl.	Onl.	Offl.	Onl.	Offl.	Onl.
Form rSCD matrix			✓					
Form cSCD matrix					✓		✓	
Filter matrix, cSCD ^f							✓	
Estimate MPS ² CD			✓	✓	✓	✓	✓	✓
Train HMMs			✓		✓		✓	
Classify with HMMs				✓		✓		✓
WordNet Similarity		✓			✓		✓	

Figure 8.6.: The different algorithms used by the techniques marked if needed online or offline. Complement classification with Definition 8.4 only classifies *unrel* and *compl*; to classify between all five document types the algorithms of $rSCD$ are also needed.

In Figure 8.6, the algorithms needed by the techniques are shown. For each technique, the algorithms needed offline (training) and online (classifying a new document) are listed. Regarding the runtime of the approaches, the calculation of the path similarities in WordNet are the most expensive part. Thus, the approaches using WordNet are much more expensive, especially if WordNet is used online during the classification. For example, calculating the values for the right plot of Figure 8.3 has a total runtime of 2.25 hours and as we see by the check mark in the first column of Figure 8.6, the calculations have to be done online. In contrast, forming the cSCD matrix used in the evaluation needs 41.1 hours, however, this is done only once and offline. Forming an rSCD matrix and training an ensemble of HMMs takes a couple of minutes. The classification used by rSCD, cSCD and cSCD^f matrices only quickly estimates cMPS²CDs and uses the pre-trained ensemble of HMMs.

In summary, using the definition of complementarity we are able to distinguish unrelated and complementary documents. However, the calculation of complementary SCDs using WordNet is slow. To allow a fast straightforward classification of all five document types, the cSCD and cSCD^f matrix combine corpora of complementary and related documents.

8.7. Interim Conclusion

If an agent is presented with a new document, it has to decide whether to extend its corpus with the new document or not—depending on the document’s type. The approach presented in this chapter enables the agent to classify a document into five types: Similar, revised, extended, unrelated, and complementary. The approach operates on the SCDs of the new document and the agent’s corpus during classification. To this end, we first give a definition of complementary SCDs and define complementary of documents based on their complementary SCDs. Second, we present the approach, which forms a cSCD matrix containing related and complementary SCDs for detecting complementarity of documents among the four other document types. In an evaluation, we demonstrate that the definition of complementarity allows an agent to separate unrelated and complementary documents. Additionally, we show that the performance using the combined SCD-word distribution matrix classifying documents of five types outperforms previous techniques not using combined SCD-word distribution matrices.

Generalizing Relations among SCDs Complementarity is only one conceivable inter-SCD relation. ReFrESH introduces factors during the propagation step of relations, too. Thus, there are more relations among SCDs to be used with the techniques introduced in this chapter.

Currently, the cSCD matrix consists of two parts representing related and complementarity documents. However, further parts can be easily appended to support further types of documents or different relations. Likewise, the cMPS²CD algorithm can be generalized for further types or relations. Presumably, the structure of similarity values needs to be changed because a third dimension cannot be added to a decimal number. However, a similarity vector representing the similarity of each type or relation would solve this issue.

Four Solutions to Four Problems In Section 3.2 we identified four problems creating our SCD-based IR agent. USEM, LESS, FrESH, and ReFrESH solve Problems I - III. Together, these techniques close the cycle of estimation, enrichment, usage, and improvement approaching text understanding. In this chapter, we provide the solution to Problem IV. As already stated, USEM can easily be combined with SEcM to estimate cSCDs matrices in an unsupervised manner.

Now that we have all the techniques, we can move on to the application part. In the second part, we describe an information system providing the SCD-based IR agent and two more exemplary use-cases of SCDs.

Part II.

Application

9. Composing an Information System using SCDs

This chapter is based on section 5 of the following publication.

- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **Unsupervised Estimation of Subjective Content Descriptions in an Information System** in *International Journal of Semantic Computing*, 2024
<https://dx.doi.org/10.1142/S1793351X24410034>

Magnus Bender developed the initial idea, implemented the information system, and wrote the manuscript. The other three authors fundamentally supervised the research by discussing ideas, proofreading the manuscript multiple times, and giving feedback. Section 5 of the journal article was taken verbatim to this chapter and then extended for the dissertation.

9.1. Introduction

In Chapter 3, we took a look at the big picture of SCDs. In doing so, we selected the example of an SCD-based IR agent. For such an agent we identified problems and provide solutions in Part I.

Hence, in this chapter we outline an SCD-based IR agent as described in Section 3.2. We integrate the IR agent in an information system accessible via a web interface. The general use-case is depicted in Figure 9.1: A user, possibly a human or another agent, supplies a corpus that represents the general field of interest of the user. The interaction with the IR agent (symbolized by the robot in Figure 9.1) is done via a computer displaying the information system. The system itself provides IR and corpus extensions using SCD. Hence, we call it the SCD-based Information System (SIS).

SIS uses the same implementation that is also being used with the evaluations of this dissertation in Part I. However, it does not implement all techniques¹, but provides the full cycle toward text understanding. First, USEM (Chapter 4) and LESS

¹Not implemented: FrESH (Chapter 6), Complementarity (Chapter 8)

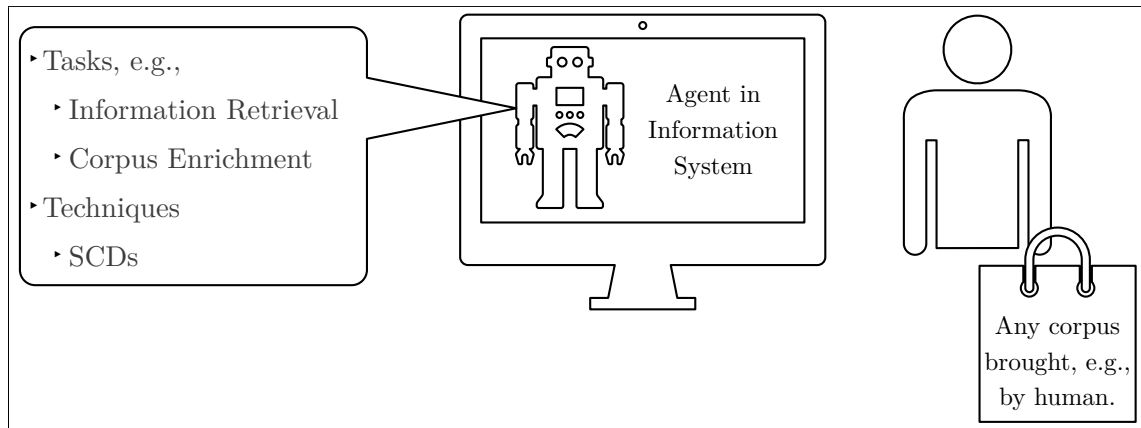


Figure 9.1.: The information system provides an web interface. In the system an SCD-based IR agent is embedded.

(Chapter 5) are used to create initial SCDs with labels for the supplied corpora. Afterwards, IR is conducted using the SCDs together with MPS²CD. Finally, ReFrESH (Chapter 7) is used to improve the SCDs with feedback.

This chapter describes SIS. Generally, a web based information system requires many features. First, the system needs an management of user accounts and of different user supplied corpora. Additionally, the system should provide a viewer for the documents of each corpus. And after that, the features that actually use SCDs will follow.


In the next sections, we first describe the basic structure of SIS. We then demonstrate how SIS can assist a user in analyzing user-submitted corpora.

9.2. Basic Structure

SIS is a web application which consists of an web interface written in HTML, CSS, and JavaScript. On the server side, SIS is written in Python using FastAPI² and runs inside a Docker container. SIS stores all corpora on the server, as well as all models consisting of the estimated SCDs.


The web interface is used to manage corpora and navigate through the contained text document and estimated SCDs. Figure 9.2 shows the index page which lists all corpora available in SIS. The upper figure shows the non-expert mode. This mode reduces the visible options available to users and mostly only shows the corpora

²<https://fastapi.tiangolo.com/>


Index of Corpora
Corpora ▾
Account
Logout
Magnus *Expert Mode*

Index of Corpora

Short name	Name	Date	Open	Status
BGB	Civil Code	14.11.2022 21:30:10	↗ Corpus	✓ More
BGBAT	Civil Code (General Part)	14.11.2022 21:30:10	↗ Corpus	✓ More
BVerfGG	Federal Constitutional Court Law	16.05.2023 18:15:49	↗ Corpus	✓ More
GG	Constitution	13.09.2022 21:35:02	↗ Corpus	✓ More


Index of Corpora
Manage Corpora
Corpora ▾
SCD Distributions ▾
Account
↗ API
Logout
Magnus *Expert Mode*

Index of Corpora

Short name	Name	Date	Open	Status
BGB	Civil Code	14.11.2022 21:30:10	↗ Corpus ↗ SCD Distribution	✓ More
BGBAT	Civil Code (General Part)	14.11.2022 21:30:10	↗ Corpus ↗ SCD Distribution	✓ More
BVerfGG	Federal Constitutional Court Law	16.05.2023 18:15:49	↗ Corpus ↗ SCD Distribution	✓ More
GG	Constitution	13.09.2022 21:35:02	↗ Corpus ↗ SCD Distribution	✓ More

Figure 9.2.: Both figures: The index page of SIS. This page shows the available corpora and provides the options to open each corpus. In right part of navigation bar, the account management is available. Upper figure: This non-expert mode provides a reduced set of available buttons. Lower figure: The expert model shows more features. E.g. it provides direct access to the SCD matrix distributions or the JSON API.

and a button to open each corpus. In the navigate bar at the top, it is possibly to manage the user account and activate the expert mode. In contrast, the expert mode, shown in the lower image, offers many more options. Again, all corpora are listed. However, the SCD matrix distributions and the JavaScript Object Notation (JSON) API is directly accessible.

The API button leads us to the next general feature of SIS. Besides the HTML based GUI, which can be used by humans, there exists a JSON API. Using the API, other applications can send queries to SIS and get machine readable responses in JSON format. This allows other agents to use SCDs without requiring special knowledge of techniques like USEM or LESS. For example, the API of SIS is used as source of information for a humanoid service robot. Thereby, the humanoid service robot provides a human-friendly way of interaction with information retrieval agents and thus brings technology closer to the people [SBM23].

Generally, SIS is capable of processing corpora in different languages, including English and German. Corpora can be supplied as plain texts, PDF documents, or law texts in XML format³. For law texts, the viewer provides some special features like highlighting and linking references⁴. The features using SCD are the same for all types of corpora. We demonstrate the features of SIS using the example of the BGB.

9.3. Working with Corpora

Let us assume, *Charlie* wants to find similarities and similar paragraphs in the German Civil Code, the BGB. First, *Charlie* opens the web interface of SIS in a web browser and uses username and password for authentication. In our case, the corpus BGB is already available in SIS and thus *Charlie* can directly choose to view the BGB. The status of a corpus is shown on the index page for each corpus, i.e., the green tick marks in Figure 9.2.

If a corpus is not available in SIS, the corpus may be imported by uploading a zip archive containing either multiple plain-text documents, PDF documents, or a single XML file³. The text documents in the zip archive then represent the corpus. After the upload, USEM estimates initial SCDs and LESS labels for the SCDs. After USEM and LESS have finished, the corpus can be viewed by *Charlie*.

³Document type definition: <https://www.gesetze-im-internet.de/dtd/1.01/gii-norm.dtd>

⁴Side note in conjunction with Chapter 8: References between law paragraphs are another type of relation among SCDs.

The screenshot shows a web interface for viewing a document. At the top, there are three navigation buttons: "← Previous", "↶ Parent", and "→ Next". Below these is the section header "§ 83" followed by the title "Foundation upon Death". The main text of the paragraph is displayed, with a yellow highlight over the sentence: "The seat of an foundation, unless otherwise provided, is the place where the administration is conducted." Below the text, there are three more navigation buttons: "← Previous", "↶ Parent", and "→ Next". At the bottom of the interface, there is a toggle switch labeled "Hover SCD Windows", which is currently turned on.

Figure 9.3.: A paragraph or page of the corpus shown in the web interface. The buttons below and above the content allow to navigate forward and backward. SCD windows are highlighted in yellow on hover.

Only for demonstration purposes, we exchange the original German texts with their English translations. SIS supports English corpora, but there exists no English XML file for the BGB.

Each content, e.g., law paragraph or page of a PDF document, is visualized as shown in Figure 9.3. Additionally to this content itself, buttons to navigate to the previous and next paragraph are available. The SCD windows are highlighted in yellow on hover, i.e., when the mouse is moved over them. Doing a double-click on a highlighted window opens the assigned SCD.

Most corpora are divided into multiple sections and thereby provide some type of structure to depict. In Figure 9.4, a larger area of the web interface visualizing corpora is shown. On the left side of the content, a table of contents is available and can be used to jump to different sections. Above the content, a small bar shows the location of the currently shown content as path through the structure of the corpus. In the upper right corner, an input box for terms to search the corpus is available. Only a simple full text search is carried out here.

Assume, *Charlie* reads different paragraphs of the BGB and is then interested in similar paragraphs to the third sentence of § 83 (highlighted in Figure 9.3). Hence, *Charlie* does a double-click on the highlighted sentence and *Charlie's* web browser opens the assigned SCD to the just double-clicked SCD window.

9.4. Working with SCDs

After doing the double-click, *Charlie's* web browser visualizes the selected SCD. The SCD was previously estimated by USEM and labelled using LESS. USEM provides three methods with different hyperparameters which result in multiple models for one corpus. Algorithm 4 provides an approach to select the best model. If *Charlie* uses SIS in the non-expert mode, *Charlie* notices nothing about the internal steps of USEM, LESS, and model selection. The selected SCD is visualized using the best model.

However, if *Charlie* uses SIS in the expert mode, SIS shows a model selection overview to *Charlie*. Such a model selection overview is shown in Figure 9.5. The overview lists all three methods of USEM with different hyperparameters. An option where nothing is merged, i.e., each sentences in one SCD, is also available. The best model identified by Algorithm 4 is marked with a star, e.g., the eighth model estimated by the greedy method with a threshold of 0.3.

Viewing SCDs in expert mode shows a navigation bar to switch between the different hyperparameters of one method of USEM. For example, Figure 9.6 shows this navigation bar for the seven models estimated by K-Means. The navigation bar enables *Charlie* to compare the SCD for one sentence of the corpus in different models. Hence, it simplifies the manual selection of the best model for the current task.

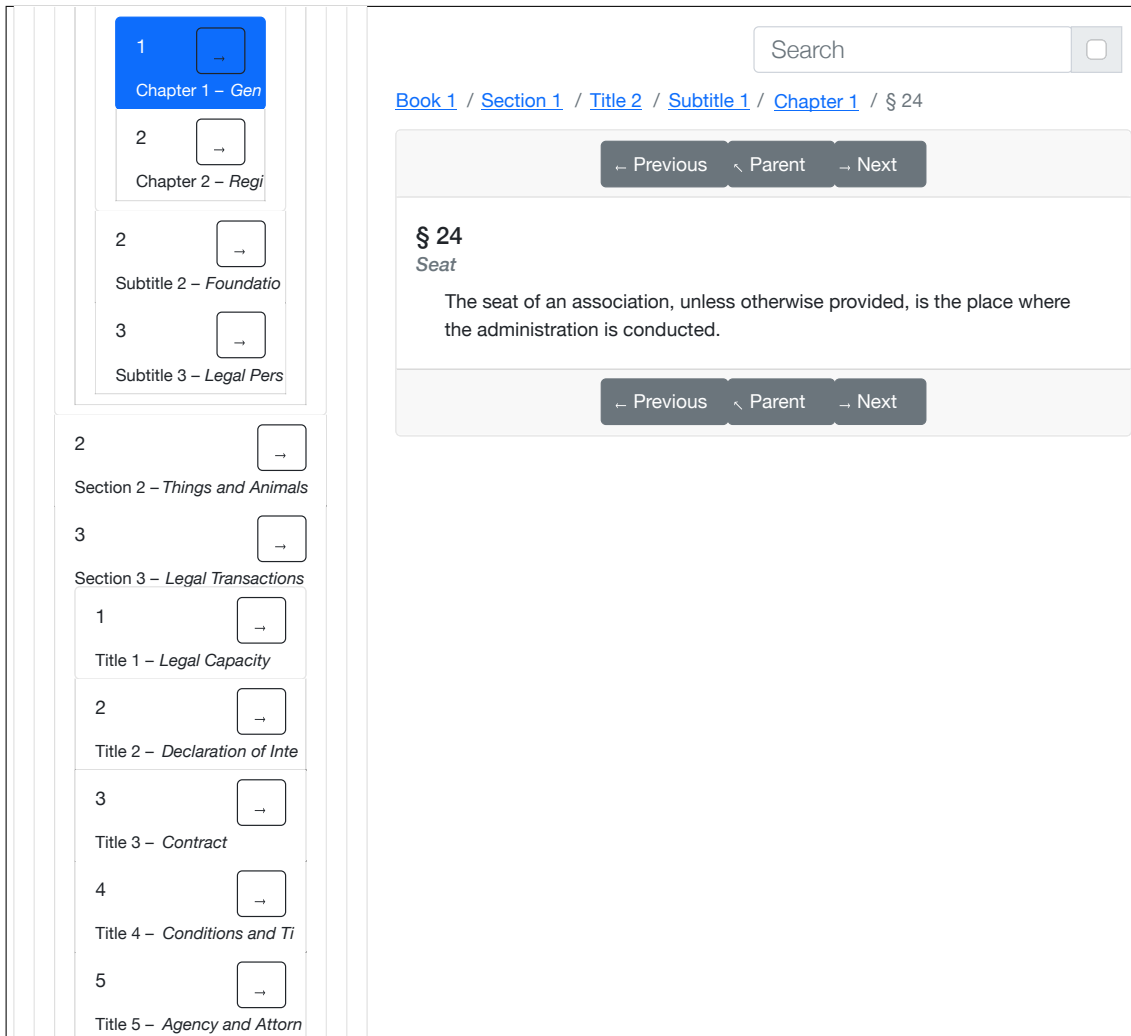


Figure 9.4.: View the documents of a corpus in SIS via the web interface. On the left side a table of contents is shown, while in the middle of the right side content is shown. In the upper right corner a full text search for the corpus is available.

No Merge	0 – Initial, no merge
Greedy by Cosine Similarity	1 – Merged identical
Cluster by KMeans	2 – Merged ≥ 0.9
Cluster by DBScan	3 – Merged ≥ 0.8
	4 – Merged ≥ 0.7
	5 – Merged ≥ 0.6
	6 – Merged ≥ 0.5
	7 – Merged ≥ 0.4
	★ 8 – Merged ≥ 0.3
	9 – Merged ≥ 0.2
	10 – Merged ≥ 0.1

Figure 9.5.: The model selection, which is only shown with the expert mode. Otherwise, Algorithm 4 selects the best model, which is also marked by a star. The different models are caused by the three methods and multiple hyperparameters of USEM.

Matrix and SCD Navigation						
0 – Initial, no merge	1 – Merged by factor 0.8	2 – Merged by factor 0.6	3 – Merged by factor 0.4	4 – Merged by factor 0.3	5 – Merged by factor 0.2	6 – Merged by factor 0.1
<div> ← Entire Matrix ← Previous <div>SCD ID 25</div> Next → </div>						

Figure 9.6.: The different models resulting from different hyperparameters of a method of USEM.

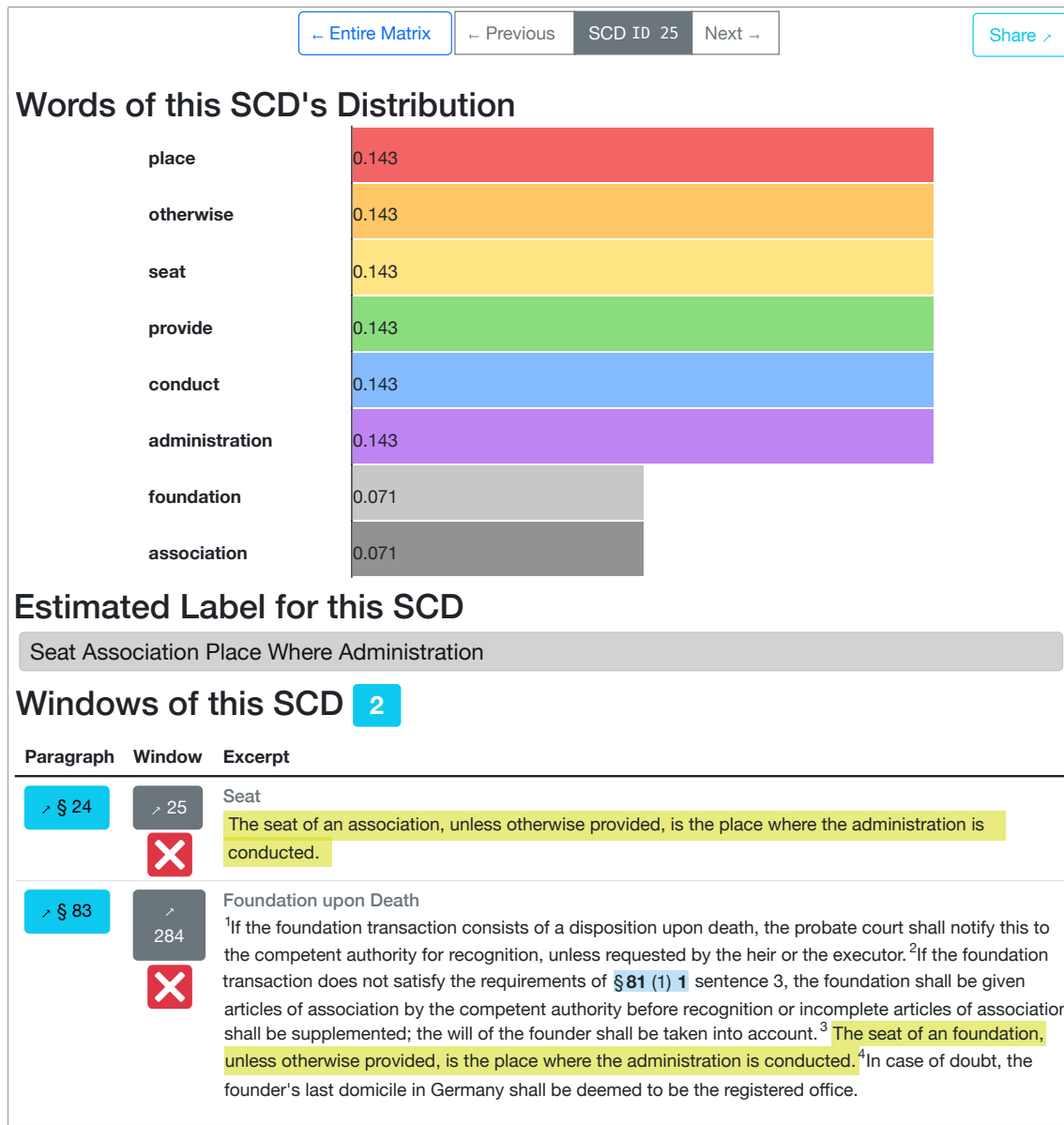


Figure 9.7.: The SCD is visualized with its word distribution and the referenced sentences. There is a list of referenced sentences, where each sentence is highlighted yellow. For each sentence the red button triggers ReFrESH on this sentence. On the top it is possible to navigate through the SCDs of the corpus and view the entire SCD matrix.

Figure 9.7 shows the visualization of an SCD. First, the most probable words of the SCD-word distribution are shown and second the estimated label by LESS. In this example, the label is truncated and the stop words are removed. Below, the referenced sentences are listed. The list does not only consist of the referenced sentences of the SCD. Each referenced sentence is shown as excerpt of the corpus together with its surrounding content while the SCD window itself is again highlighted yellow. Showing the surrounding sentences is more human friendly and allows *Charlie* to grasp the context of the referenced and similar sentences more quickly. Thus, *Charlie* can identify the most relevant sentences for *Charlie's* information need and choose to open a corpus' paragraph or SCD window using the blue or gray buttons left of the excerpt.

If *Charlie* detects a sentence which does not match the SCD, the red button with the cross triggers ReFrESH on this sentence. ReFrESH updates the SCD and reassigns the sentences to new or better matching SCDs. If *Charlie* is still not satisfied with the result of ReFrESH, *Charlie* can click the button triggering ReFrESH again. SIS also provides the possibility to reset the changes made by ReFrESH, i.e., *Charlie* may try ReFrESH without risks.

Let us assume, an SCD matches *Charlie's* information need and *Charlie* wants to share this SCD with others. In the upper right corner of Figure 9.7 SIS displays a share button. Clicking this button copies a permanent Uniform Resource Locator (URL) to the visualization of the SCD into *Charlie's* clipboard. Items and views of SIS can be cited as hyperlink using this URL.

However, it could also be the case that the SCD does not fulfill *Charlie's* information need and all sentences are suitable for the SCD, i.e., the red ReFrESH button does not help. In this case, it is also possible to browse all the SCDs estimated by USEM. The entire SCD matrix, the previous, and next SCD can be viewed with the upper buttons.

In Figure 9.8, the visualization of an estimated SCD matrix is shown. As there are many SCDs in one matrix, there are multiple pages listing all the SCDs. A pagination bar at the top allows to move from one page to the next. Above the pagination, some statistics of the matrix are shown. Below the pagination, a short overview of each SCD is shown: The overview consists of the top 10 words of the SCD-word distribution, the label estimated by LESS, the ids of the referenced SCD windows, and the number of referenced windows. It is possible to open each SCD with the blue button on the left and the referenced windows using the gray buttons on the right below the label.

However, there is a huge amount of SCDs and *Charlie* can hardly go through all the pages of SCDs to identify the most relevant SCD for *Charlie's* information need. Thus, SIS provides a MPS²CD based search through all the SCDs.

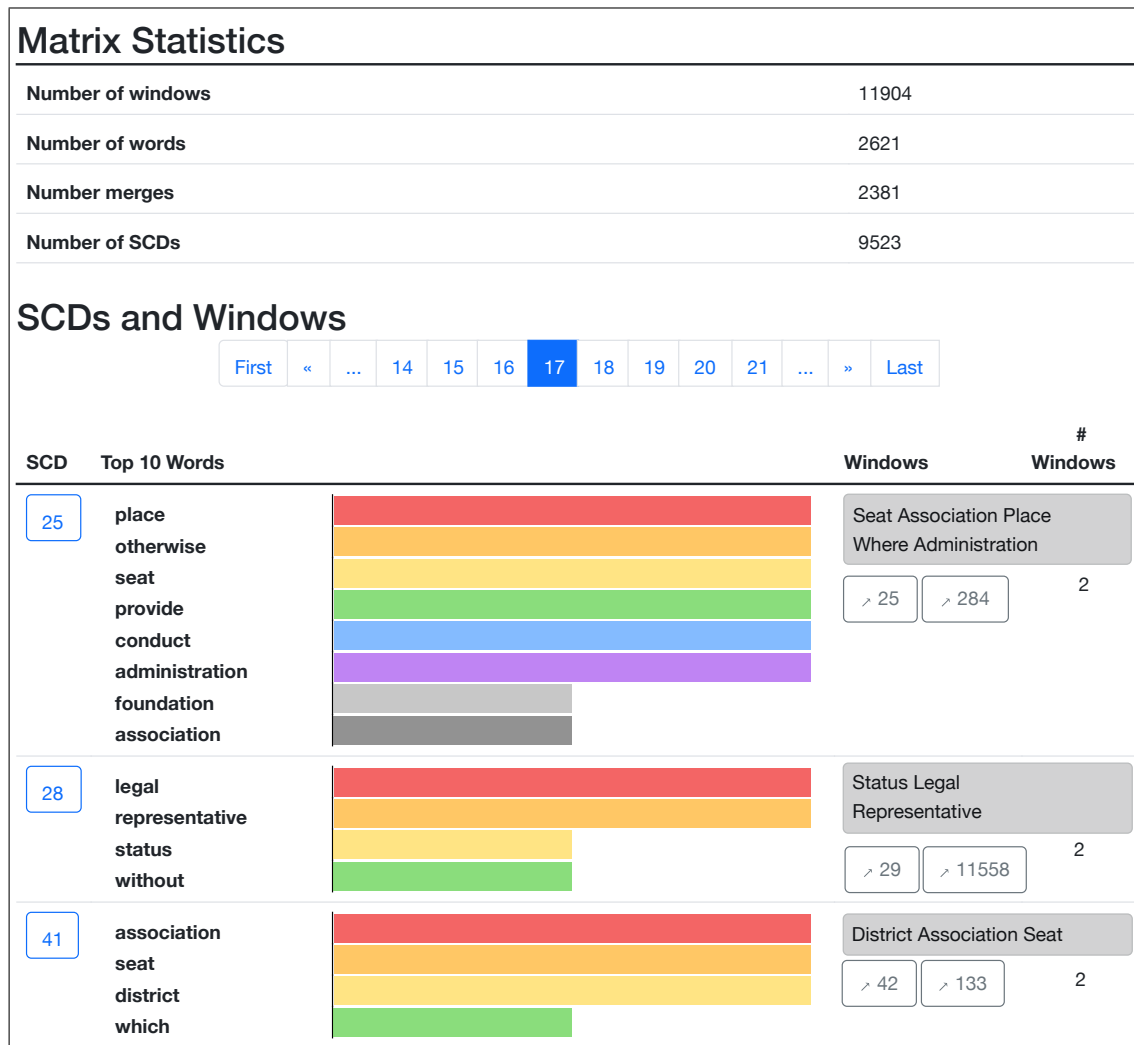


Figure 9.8.: The SCDs of an SCD matrix are shown across multiple pages with a navigation between the pages on top. Each SCD is depicted by its word distribution, a label, the ids of the referenced SCD windows, and the number of referenced windows. There is a button to view each SCD on the left side and some statistics on top.

9. Composing an Information System using SCDs

The MPS²CD algorithm estimates a most probably suited SCDs for a single previously unseen text document. Here, the unseen text document is the user supplied query representing the user's information need. For this search query, the most probably suited SCDs are identified and displayed to the user. Thus, SIS provides a powerful search using the SCDs of the corpus.

In Figure 9.9, the MPS²CD based search is shown. User *Charlie* types the query into the textbox on top and hits the search button. Then, SIS uses MPS²CD and shows a list of most probably suited SCDs in descending order of similarity. For each MPS²CD the similarity score and the label is shown.

Charlie can unfold each SCD in the list to get the referenced sentences of the SCD. Again, for each referenced sentence an excerpt of the corpus with its surrounding content is shown while the SCD window itself is highlighted yellow. Thus, *Charlie* can identify the best fitting SCD and view the SCD or the content by clicking the buttons. The MPS²CD based search provides a button to get a hyperlink for sharing, too.

Overall, SIS provides an interactive and visualized interface for humans to interact with an SCD-based IR. Especially SIS runs unsupervised and with a small computational footprint. It allows users to import their corpora and browse all the documents. Labels help the users to grasp the estimated SCDs which represent the concepts and locations across the corpus. Additionally, the MPS²CD based search allows the users to formulate a query using their own words. Furthermore, users can create improved versions of initially estimated SCDs using ReFrESH.

Search SCDs (using MPS²CD)

Share

The seat of an association shall be the place of its administration.

Search

Results

25

0.7893522173763263

Seat Association Place Where Administration

Paragraph

Window

Excerpt

24

25

Seat

The seat of an association, unless otherwise provided, is the place where the administration is conducted.

83

284

Foundation upon Death

¹If the foundation transaction consists of a disposition upon death, the probate court shall notify this to the competent authority for recognition, unless requested by the heir or the executor. ²If the foundation transaction does not satisfy the requirements of §81 (1) 1 sentence 3, the foundation shall be given articles of association by the competent authority before recognition or incomplete articles of association shall be supplemented; the will of the founder shall be taken into account. ³The seat of an foundation, unless otherwise provided, is the place where the administration is conducted. ⁴In case of doubt, the founder's last domicile in Germany shall be deemed to be the registered office.

41

0.49613893835683387

District Association Seat

Figure 9.9.: The MPS²CD based search fetches the most similar SCDs based on a user supplied query. The query is inserted into the textarea on top and the similar SCDs are shown together with a similarity score and their referenced sentences.

121

10. SCDs in Further Domains

The Section 10.2 of this chapter is based on section 6 of the following publication:

- Thomas Asselborn, Sylvia Melzer, Said Aljoumani, Magnus Bender, Florian Andreas Marwitz, Konrad Hirschler and Ralf Möller: **Fine-tuning BERT Models on Demand for Information Systems Explained Using Training Data from Pre-modern Arabic** in *Proceedings of the Humanities-Centred AI (CHAI) Workshop at KI2023, 46th German Conference on Artificial Intelligence, 2023*
<https://ceur-ws.org/Vol-3580/paper5.pdf> (Slides: <https://dx.doi.org/10.25592/uhhfdm.13423>)

Section 6 of the workshop paper was taken to Section 10.2 and then adapted for the dissertation. For Section 6 of the workshop paper: Magnus Bender developed the initial idea and mainly wrote the section. Florian Marwitz lectured and extended the section. Ralf Möller fundamentally supervised the research by discussing ideas, proofreading the section, and giving feedback.

10.1. Introduction

An SCD-based IR agent embedded in an information system like SIS is one use-case of SCDs. In this dissertation, the IR agent provides the read thread which connects all the chapters and techniques. However, SCDs can be used in different use-cases, too.

This chapter shows two possible applications of SCDs in wider domains. First, we do not use SCDs as main model, but as post-processing technique to optimize responses before they are sent to users. For this we especially need USEM and MPS²CD.

Second, we consider how the SIS can be integrated in other applications. Especially it is important to keep the barriers of using SIS as low as possible.

10.2. Humanities Aligned Chatbot

Often IR services are available through a web interface provided by an information system. Thus, users need to fiddle around with different interfaces and each interface needs to be created for the specific tasks of a user. Typically, scholars in the humanities work with large corpora that they need to analyze. Besides providing different interfaces per task, the scholars need to get to know each new interface. Instead, we would like to build a system with a more general interface. The system should allow the scholars to supply their corpora easily and to interact in a more natural way.

As solution, we outline ChatHA, a Humanities Aligned Chatbot. We adopt the mechanism of fine-tuning an LLM, e.g., version 4 of GPT [RN18]. Fine-tuning LLMs is a step to adapt an already pre-trained LLM to a specific task or corpus. A pre-trained LLM is already generally trained to process and generate language, but lacks a specific task. ChatHA runs the fine-tuning of the pre-trained LLM for each user supplied corpus. Afterwards, the fine-tuned models know the corpus well and still has the ability to process and generate language.

The fine-tuned model can be used to provide a chatbot that can answer natural language questions about the corpus. In contrast to publicly available chatbots, ChatHA gets fine-tuned on the specific data in the user supplied corpus. Hence, ChatHA is able to provide detailed answers based on the available data. Additionally, the ability to process natural language questions and providing the answers the same way lowers the barriers for the humanities scholars to use the system. It is not necessary to create a graphical user interface for a task, as each task can be sent as a textual question to ChatHA.

Even the extraction of the corpus can be done with the help of an LLM. If the user supplies the text documents of the corpus as files, these files can be used for fine-tuning in the next step. However, a user may specify a more general description of the corpus, e.g., provide an hyperlink to a web site with the text documents. In this case, ChatHA needs to download the corpus before it can start the fine-tuning. AutoGPT [YYH23] is a chatbot for more sophisticated tasks. The idea is, that AutoGPT automatically splits the question into multiple smaller ones. Thereby, each question is directly answered by AutoGPT itself. In doing so, it is for example possible to automatically crawl the web and download the user specified corpus.

Using AutoGPT it is also possible to do corpus extension and search for information which goes beyond the information available in the corpus. For example, corresponding academic publications and further web resources about the topic could be automatically included. Generally, AutoGPT can be used to answer more challenging questions that require deeper understanding and expressiveness.

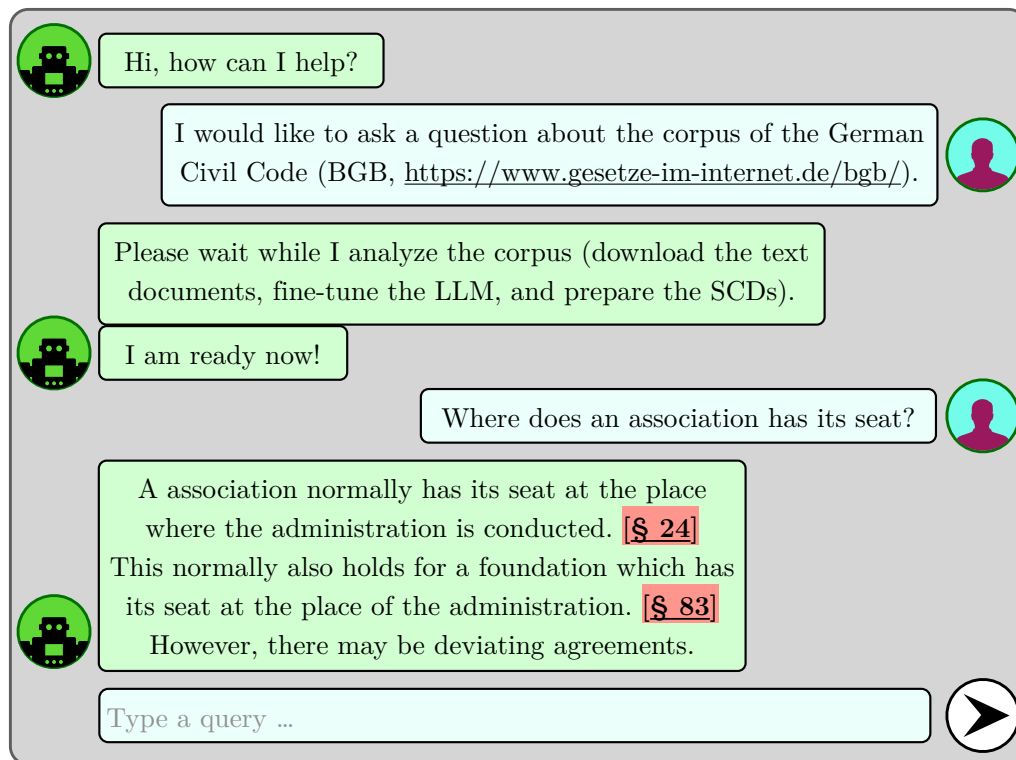


Figure 10.1.: A fictitious conversation with ChatHA about the BGB as mockup. It uses the same examples already used with the SIS and MPS²CD in Figure 9.9.

However, a system like ChatHA built automatically and with less supervision also implies some issues: LLMs have no *true* understanding of the corpus and its content, they try to combine the best answer based on texts they processed during training. Therefore, LLMs are prone to hallucinations, erroneous answers invented by the LLM, and do not cite their sources. To combat this issues, we do not output the raw LLM output during question answering, but rather post-process it to include citations. The obtained result is then displayed to the user and the citations allow the user to validate the answer.

An example for a possible conversation with ChatHA is shown as mockup in Figure 10.1. The user sends a question including a hyperlink to the corpus. ChatHA now needs to download this corpus, in the example the BGB, and fine-tune the LLM. Additionally, ChatHA prepares SCDs for the post-processing of the raw LLM output. When ChatHA is ready for questions about the BGB, the user asks a question (the question is similar to the query in Figure 9.9). ChatHA answers with the post-processed answer of the fine-tuned LLM. Thereby, it uses SCDs to add citations to text documents in the corpus, e.g., §§ 24, 83.

We now describe this post-processing and the overall workflow in more detail: First, we choose some pre-trained LLM. We opt for using a pre-trained version to include basic natural language understanding and general query answering. Second, the user, in our case a humanities scholar, chooses on which types of texts the LLM should be fine-tuned on. The texts do not have to be in English, e.g., Arabic or Tamil [AMA⁺23, BBG⁺21b] are also possible. This step composes the corpus for our LLM and ensures the alignment for humanities of the fine-tuned LLM. Third, we fine-tune the LLM with the selected data and create the chatbot by this step. However, we still have the issue of hallucinations and missing citations.

As a solution, we apply SCDs. Using the SCD matrix, an SCD can be identified by the MPS²CD algorithm for any new and unseen sentence. MPS²CD identifies the most suitable SCD from the set of known SCDs associated with the text documents. Hence, using MPS²CD it is possible to create a link from a new and unseen sentence to an SCD and all sentences this SCD is associated with.

Coming back to ChatHA, we lack SCDs on the corpus used for fine-tuning. Using USEM we add these SCDs to the corpus used for fine-tuning—each sentence gets one SCD. Afterwards, each SCD represents a topic or concept mentioned in the corpus and all sentences about each topic or concept belong to the same SCD. Thus, our SCDs represents the various topics or concepts in the corpus and the sentences that mention them.

Using SCDs we can solve the issue of hallucinations and missing citations in the output of the LLM. For each sentence in the output, MPS²CD identifies an SCD from the corpus. In doing so, a link from the output of the LLM to the SCDs of the corpus and further on to sentences of the corpus is created. These links can now be used as citations shown in the output pointing to relevant sentence in the corpus used for fine-tuning.

Finally, ChatHA is ready to be used: A humanities scholar inputs a question about a previously supplied corpus using natural language. This question is first sent to the LLM and its output is post-processed in the following way: We apply MPS²CD on the raw output to identify an SCD for each sentence. Sentences for which no SCD is found may be hallucinations and are omitted because there is no evidence of SCDs. The processed output is then displayed to the user alongside with the SCDs for each sentence. For each sentence and SCD, ChatHA may offer the possibility to view this SCD in SIS or to open each of the sentences in the corpus used for fine-tuning which are associated to the same SCD. Additionally, if further visualization is available for an SCD or a sentence, ChatHA offers to visualize it in the corresponding information system.

All in all, ChatHA can be used to query the research data repository for research

tasks in the humanities. The output includes citations, so we not only reduce hallucinations, but also give pointers for a more detailed look.

10.3. Research Data Repository Integration

An information system helps a user browse a corpus of documents. An embedded agent in the system adds the possibility to interact with the corpus. Generally, there are many different information system available and each system provides a different set of features. Thus, for each type of document in a dataset some information systems are *good* choices and others are not. For example, SIS works with any corpus of PDF or plain text documents and especially well with law texts.

Often, datasets are stored in repositories. In the context of research, these are called Research Data Repositories (RDRs). InvenioRDM¹ is a software to deploy RDRs. Similarly, Kaggle² is a service providing an RDR by Google.

Let us assume a user needs to collect information about some topic. The user will start with browsing through the datasets in an RDR. If a dataset seems to be relevant for the topic, the user needs to download the dataset and view it locally on the user's computer. However, downloading a dataset just to take a first look adds a lot of unnecessary overhead and raises the barriers to using a dataset. A solution would be to integrate previews and information systems into the RDR, such that a user is able to directly open the dataset in an information system supporting this type of data.

This solution is similar to the ideas in [SM23, AMA⁺23]. Schiff and Möller [SM23] describe the integration of a viewer including a processor of so-called critical editions into InvenioRDM. In [AMA⁺23], Asselborn et al. describe a fine-tuning on demand process. A user can request a LLM fine-tuned on a corpus from the RDR. The latter may also be used to fine-tune LLMs for ChatHA right from the RDR.



In our case, we follow the goal to integrate SIS in an RDR like InvenioRDM, i.e, we create SIS+RDR. Thereby, SIS+RDR should require only small changes to the RDR. InvenioRDM only needs to display a button to open SIS beneath each supported dataset and this button is just a hyperlink. Figure 10.2 shows a mockup of how an “Open in Information System” button might look. This button would be shown with any dataset of the RDR containing PDF documents, plain texts, or law texts in XML format.

¹<https://inveniosoftware.org/products/rdm>

²<https://www.kaggle.com/datasets>

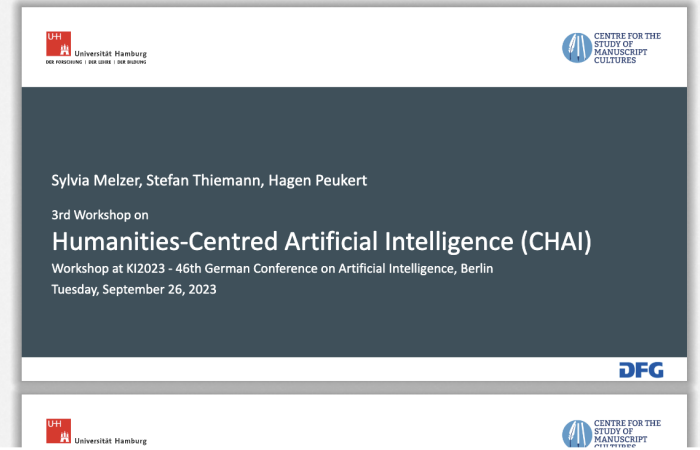
10. SCDs in Further Domains

UNIVERSITÄT HAMBURG







The KI2023 workshop – Humanities-Centred AI was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2176 'Understanding Written Artefacts: Material, Interaction and Transmission in Manuscript Cultures', project no. 390893796.

Preview



Files (13.2 MB)

[Open in Information System](#)

Name	Size	
00_CHAI2023_Agenda.pdf	203.2 kB	 Preview  Download
md5:0d454edbbf82a339362929e2ce23b039 ?		
presentation1.pdf	1.4 MB	 Preview  Download

Versions

Version 1

10.25592/uhhfdm.13423

Sep 26, 2023

Cite all versions? You can cite all versions by using the DOI [10.25592/uhhfdm.13423](https://doi.org/10.25592/uhhfdm.13423). This DOI represents all versions, and will always resolve to the latest one.

Cite record as

Sylvia Melzer, Stefan Thiemann, & Hagen Peukert. (2023, September). 3rd Workshop on Humanities-Centred Artificial Intelligence (CHAI 2023). <http://doi.org/10.25592/uhhfdm.13423>

Start typing a citation style...

Export

[BibTeX](#) [CSL](#) [DataCite](#) [Dublin Core](#) [JSON](#) [JSON-LD](#) [MARCXML](#) [Mendeley](#)

Figure 10.2.: Mockup integrating an “Open in Information System” button into an RDR. In this example, we added the button to <https://www.fdr.uni-hamburg.de/record/13423>.

Clicking the button opens the web application of SIS. In this process, the dataset needs to be passed to SIS. SIS provides an import API for this purpose. The import API is accessed by the user via a hyperlink created by the RDR. The hyperlink itself contains a hyperlink to the dataset to import and optionally name, language, and type of the dataset. Based on this information, SIS checks if the dataset is already available. If it is not available, SIS runs USEM and LESS and estimates SCDs with labels for the dataset as new corpus. When the SIS has finished the estimation, the user can use SIS as described in Chapter 9.

To round off the import API, SIS also provides an export functionality of corpora together with their SCDs. Hence, the user may use ReFrESH to improve the SCDs of a corpus and export them afterwards. The export file can then be added to the RDR and the RDR will again display an “Open in Information System” button. The import and export functionalities enable collaboration between multiple users of SIS+RDR: One user uploads an export with improved SCDs to the RDR and another user imports this corpus to SIS and further improves it. Afterwards, the other user may again upload the new improved version to the RDR and so on.

SIS+RDR lowers the barriers using SCDs and datasets in the RDR. A user only needs a computer with a web browser. All the processing is done on the server-side by the RDR and SIS. SCDs follow the idea of lean computing and are satisfied with off-the-shelve hardware, i.e., it is possible to host SIS for multiple users with a reasonable amount of computational resources.

In this second part, we described our information system providing the SCD-based IR agent, its integration in an RDR, and ChatHA. Next, we conclude the dissertation with a summary of contributions and an outlook.

11. Conclusion

Natural language is the most intuitive way of human communication. The techniques presented in this dissertation provide a next step toward text understanding in the sense of NLP. We use the example of an IR agent and solve four problems composing such an agent. Together the techniques build a cycle of estimating, enriching, using, and improving SCDs associated with a corpus of text documents. Iteration after iteration on the cycle, the SCDs improve incrementally toward the top of a spiral.

An SCD-based IR agent using the presented techniques is embedded in SIS. SIS makes it easy for human users to use SCDs with their corpora and corpora available in RDRs. Furthermore, SCDs and a subset of the techniques from this dissertation are applied in ChatHA.

In the final two sections, we first summarize the contributions of the dissertation and then present possible future work.

11.1. Summary of Contributions

We start the dissertation with a big picture providing an overview about SCDs. In the first part, we then present five new techniques using SCDs. In the second part, we describe applications using SCD and the previously presented techniques.

Part I Five chapters provide the theoretical foundation of the dissertation. We start with USEM (Chapter 4) which estimates initial SCDs for any corpus. Thus, our IR agent is able to work with user supplied corpora.

However, the newly estimated SCDs lack labels or descriptions. Hence, it is difficult for human users to grasp the topic or concept of an SCD. Therefore, LESS (Chapter 5) estimates labels for the newly estimated SCDs.

Afterwards, the IR agent is ready to respond to queries. However, users may provide feedback about faulty sentences or associations of SCDs and sentences. The agent then needs to incorporate the feedback. We have to distinguish between a sentence of faulty content, which needs to be removed entirely from the corpus, and a faulty association of sentences and SCD, where the association needs to be

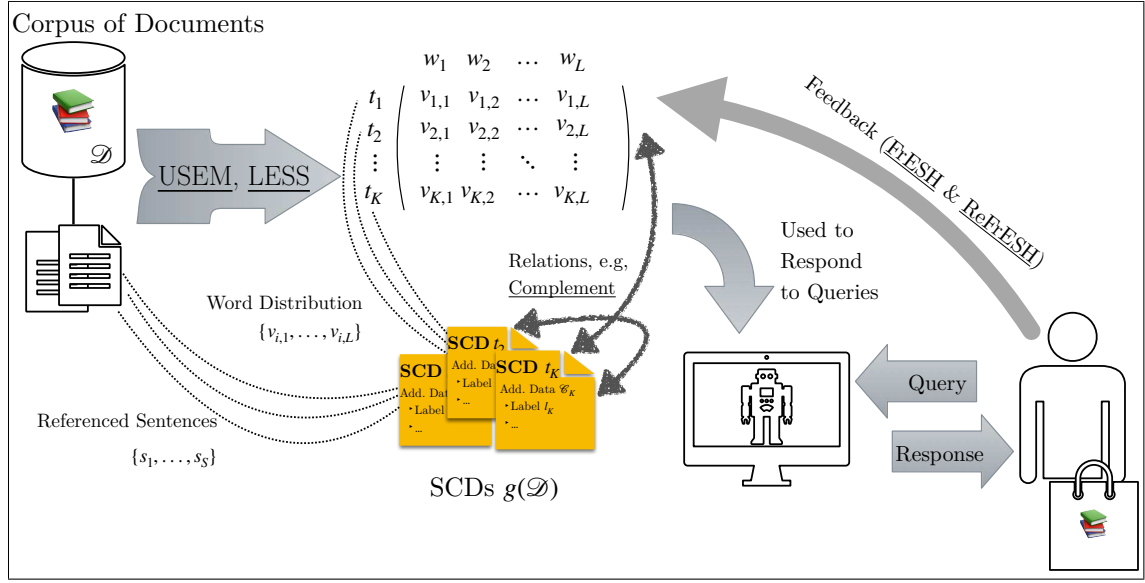


Figure 11.1.: Overview of an SCD with different relations and technique used by the SCD-based IR agent. The underlined technique presented in this dissertation.

updated. To both cases, we present a technique to incorporate the feedback, i.e., FrESH (Chapter 6) to entirely remove faulty sentences and ReFrESH (Chapter 7) to update associations and preserve relations.

Finally, we deep-dive into relations among SCDs (Chapter 8), i.e., inter-SCDs relations between two SCDs. First, we define complementarity among SCDs. Afterwards, we use complementarity as an example relation and introduce cSCD matrices, SEcM, and, cMPS²CD. These techniques are enhanced versions to work with SCDs having inter-SCD relations.

Figure 11.1 shows an overview of the different techniques presented in Part I. Each underlined term corresponds to one contribution of the dissertation. On the top left side, USEM and LESS are shown. In the middle, multiple SCDs are depicted together with the SCD matrix and the intra-SCD relations to the referenced sentences and matrix rows. Inter-SCD relations, e.g., complementarity, are also added to the figure. On the bottom right side, the IR agent is shown together with an arrow representing feedback, i.e., FrESH and ReFrESH.

Part II In two chapters, we describe three applications of SCDs. We start with SIS (Chapter 9), an information system providing all the features of an SCD-based IR agent via a web interface. We describe all required features and illustrate how

such a system could look like. Additionally, we propose a way to connect SIS to an RDR, i.e., SIS+RDR (Section 10.3).

It is not necessary that SCDs always provide the main model for an NLP task. ChatHA (Section 10.2) is a chatbot based on an LLM and SCDs are used to post-process the LLM's raw output. During the post-processing, citations are added to the answers and hallucinations are mitigated.

11.2. Outlook

Moving forward from this dissertation, there are many interesting topics for future work. We consider the following three.

Feedback Planning and Reinforcement Learning FrESH and ReFrESH assume that one sentence needs to be removed or reassigned to a different SCD. However, human feedback is often not as clear. Additionally, there are other reasons why something may be faulty from the point of a user, e.g., the response does not match the information need or the query is worded incorrectly.

An agent dealing with such unclear feedback, first needs to classify the feedback and then select an appropriate action to change its behavior. To get reliable evidence before conducting an action, the agent should plan the next actions and value their utility. All in all, such a strategy allows the agent to do reinforcement learning. Based on human feedback and the other actions of the users, the agent is able to update its models and plans to be more beneficial for users.

SCDs and Probabilistic Graphical Models The SCD matrix is a generative model and each row represents a probability distribution of words. Additionally, each SCD has references to multiple sentences in the corpus and relations including factors to other SCDs. With these items we have the main parts needed to assemble a probabilistic graphical model or factor graph. Automatically creating such a model or graph from a corpus of text documents would be a significant step and important connection of different fields of research.

Verifiability SCDs and the techniques used with SCDs are trackable, i.e., steps can be retraced by a human. Additionally, the SCD matrix has a clearly defined meaning. Hence, a response generated using SCDs can be explained by the model and thus also be verified. This verifiability makes SCDs usable in areas where an explanation or verification of responses is needed.

The use-case is similar to ChatHA: SCDs are not able to generate answers of natural language, but can be used to post-process, i.e., validate, the raw LLMs' outputs.

A. Appendix

Bibliography

- [ABK⁺07] AUER, Sören ; BIZER, Christian ; KOBILAROV, Georgi ; LEHMANN, Jens ; CYGANIAK, Richard ; IVES, Zachary: DBpedia: A Nucleus for a Web of Open Data. In: *The Semantic Web*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007. – ISBN 978-3-540-76298-0, p. 722–735
- [AJPM15] ANGELI, Gabor ; JOHNSON PREMKUMAR, Melvin J. ; MANNING, Christopher D.: Leveraging Linguistic Structure For Open Domain Information Extraction. In: *Proceedings of the Association of Computational Linguistics (ACL)* (2015), 344–354. <https://doi.org/10.3115/v1/P15-1034>
- [AMA⁺23] ASSELBORN, Thomas ; MELZER, Sylvia ; ALJOUMANI, Said ; BENDER, Magnus ; MARWITZ, Florian A. ; HIRSCHLER, Konrad ; MÖLLER, Ralf: Fine-tuning BERT Models on Demand for Information Systems Explained Using Training Data from Pre-modern Arabic. In: *Proceedings of the Workshop on Humanities-Centred Artificial Intelligence (CHAI 2023)*, CEUR Workshop Proceedings, 2023, 38–51
- [Bau72] BAUM, Leonard E.: An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes. In: SHISHA, Oved (Ed.): *Inequalities III: Proceedings of the Third Symposium on Inequalities*. University of California, Los Angeles : Academic Press, 1972, p. 1–8
- [BBG⁺21a] BENDER, Magnus ; BRAUN, Tanya ; GEHRKE, Marcel ; KUHR, Felix ; MÖLLER, Ralf ; SCHIFF, Simon: Identifying and Translating Subjective Content Descriptions Among Texts. In: *International Journal of Semantic Computing* 15 (2021), no. 4, 461–485. <https://dx.doi.org/10.1142/S1793351X21400122>
- [BBG⁺21b] BENDER, Magnus ; BRAUN, Tanya ; GEHRKE, Marcel ; KUHR, Felix ; MÖLLER, Ralf ; SCHIFF, Simon: Identifying Subjective Content Descriptions among Text. In: *Proceedings of the 15th IEEE International Conference on Semantic Computing (ICSC-21)* (2021). <https://doi.org/10.1109/ICSC50631.2021.00008>

- [BKB22] BENDER, Magnus ; KUHR, Felix ; BRAUN, Tanya: To Extend or not to Extend? Complementary Documents. In: *16th IEEE International Conference on Semantic Computing, (ICSC 2022), Virtual, January 26-28 (2022)*, 17-24. <https://doi.org/10.1109/ICSC52841.2022.00011>
- [BL06] BLEI, David M. ; LAFFERTY, John D.: Dynamic Topic Models. In: *Proceedings of the 23rd International Conference on Machine Learning* (2006), p. 113–120. <http://dx.doi.org/10.1145/1143844.1143859>. – DOI 10.1145/1143844.1143859. ISBN 1595933832
- [BLB16] BHATIA, Shraey ; LAU, Jey H. ; BALDWIN, Timothy: Automatic Labelling of Topics with Neural Embeddings. In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan : The COLING 2016 Organizing Committee, December 2016, 953–963
- [BMR⁺20] BROWN, Tom B. ; MANN, Benjamin ; RYDER, Nick ; SUBBIAH, Melanie ; KAPLAN, Jared ; DHARIWAL, Prafulla ; NEELAKANTAN, Arvind ; SHYAM, Pranav ; SASTRY, Girish ; ASKELL, Amanda ; AGARWAL, Sandhini ; HERBERT-VOSS, Ariel ; KRUEGER, Gretchen ; HENIGHAN, Tom ; CHILD, Rewon ; RAMESH, Aditya ; ZIEGLER, Daniel M. ; WU, Jeffrey ; WINTER, Clemens ; HESSE, Christopher ; CHEN, Mark ; SIGLER, Eric ; LITWIN, Mateusz ; GRAY, Scott ; CHESS, Benjamin ; CLARK, Jack ; BERNER, Christopher ; MCCANDLISH, Sam ; RADFORD, Alec ; SUTSKEVER, Ilya ; AMODEI, Dario: Language Models are Few-Shot Learners. (2020). <https://arxiv.org/abs/2005.14165>
- [BNJ03] BLEI, David M. ; NG, Andrew Y. ; JORDAN, Michael I.: Latent Dirichlet Allocation. In: *Journal of Machine Learning Research* 3 (2003), 993–1022. <http://jmlr.org/papers/v3/blei03a.html>
- [BPSW70] BAUM, Leonard E. ; PETRIE, Ted ; SOULES, George ; WEISS, Norman: A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. In: *The annals of mathematical statistics* 41 (1970), no. 1, p. 164–171
- [CGR⁺17] COLLARANA, Diego ; GALKIN, Mikhail ; RIBÓN, Ignacio T. ; VIDAL, Maria-Esther ; LANGE, Christoph ; AUER, Sören: MINTE: Semantically integrating RDF graphs. In: *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, WIMS 2017, Amantea, Italy, June 19-22, 2017*, 2017, p. 22:1–22:11
- [DCLT19] DEVLIN, Jacob ; CHANG, Ming-Wei ; LEE, Kenton ; TOUTANOVA, Kristina: BERT: Pre-training of Deep Bidirectional Transformers

- for Language Understanding. (2019). <https://arxiv.org/abs/1810.04805>
- [DLR77] DEMPSTER, Arthur P. ; LAIRD, Nan M. ; RUBIN, Donald B.: Maximum likelihood from incomplete data via the EM algorithm. In: *Journal of the royal statistical society. Series B (methodological)* (1977), p. 1–38
- [EGGM18] ELNAGGAR, Ahmed ; GEBENDORFER, Christoph ; GLASER, Ingo ; MATTHES, Florian: Multi-Task Deep Learning for Legal Document Translation, Summarization and Multi-Label Classification. In: *Proceedings of the 2018 Artificial Intelligence and Cloud Computing Conference*. New York, NY, USA : Association for Computing Machinery, 2018 (AICCC '18). – ISBN 9781450366236, p. 9–15
- [EK SX96] ESTER, Martin ; KRIEGEL, Hans-Peter ; SANDER, Jörg ; XU, Xiaowei: A density-based algorithm for discovering clusters in large spatial databases with noise. (1996), p. 226–231
- [GGS⁺20] GEHMAN, Samuel ; GURURANGAN, Suchin ; SAP, Maarten ; CHOI, Yejin ; SMITH, Noah A.: RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online : Association for Computational Linguistics, November 2020, p. 3356–3369
- [GGVZ19] GINART, Antonio A. ; GUAN, Melody Y. ; VALIANT, Gregory ; ZOU, James: *Making AI Forget You: Data Deletion in Machine Learning*. 2019
- [HBT96] HU, Jianying ; BROWN, Michael K. ; TURIN, William: HMM based online handwriting recognition. In: *IEEE Transactions on pattern analysis and machine intelligence* 18 (1996), no. 10, p. 1039–1045
- [HEGM13] HINDLE, Abram ; ERNST, Neil A. ; GODFREY, Michael W. ; MYLOPOULOS, John: Automated topic naming. In: *Empirical Software Engineering* 18 (2013), no. 6, p. 1125–1155
- [Hel09] HELLINGER, Ernst: Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen. In: *Journal für die reine und angewandte Mathematik* (1909), p. 210–271
- [HKH⁺01] HATZIVASSILOGLU, Vasileios ; KLA VANS, Judith L. ; HOLCOMBE, Melissa L. ; BARZILAY, Regina ; KAN, Min-Yen ; MCKEOWN, Kathleen: SimFinder: A flexible clustering tool for summarization. (2001)
- [HS97] HOCHREITER, Sepp ; SCHMIDHUBER, Jürgen: Long Short-Term Memory. In: *Neural Comput.* 9 (1997), November, no. 8, 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>. – ISSN 0899–7667

- [HSE11] HADANO, Masashi ; SHIMADA, Kazutaka ; ENDO, Tsutomu: Aspect Identification of Sentiment Sentences Using A Clustering Algorithm. In: *Procedia - Social and Behavioral Sciences* 27 (2011), 22-31. <https://doi.org/10.1016/j.sbspro.2011.10.579>. – ISSN 1877-0428. – Computational Linguistics and Related Fields
- [ISCZ21] IZZO, Zachary ; SMART, Mary A. ; CHAUDHURI, Kamalika ; ZOU, James Y.: Approximate Data Deletion from Machine Learning Models. In: *International Conference on Artificial Intelligence and Statistics*, 2021
- [KBBM19] KUHR, Felix ; BRAUN, Tanya ; BENDER, Magnus ; MÖLLER, Ralf: To Extend or not to Extend? Context-specific Corpus Enrichment. In: *Proceedings of AI 2019: Advances in Artificial Intelligence* (2019), 357–368. https://doi.org/10.1007/978-3-030-35288-2_29. ISBN 978-3-030-35288-2
- [KBBM20] KUHR, Felix ; BENDER, Magnus ; BRAUN, Tanya ; MÖLLER, Ralf: Augmenting and Automating Corpus Enrichment. In: *Int. J. Semantic Computing* 14 (2020), no. 2, 173–197. <https://doi.org/10.1142/S1793351X20400061>
- [KBBM21] KUHR, Felix ; BENDER, Magnus ; BRAUN, Tanya ; MÖLLER, Ralf: Context-specific Adaptation of Subjective Content Descriptions. In: *15th IEEE International Conference on Semantic Computing, (ICSC 2021), Laguna Hills, CA, USA, January 27-29* (2021), 134–139. <https://dx.doi.org/10.1109/ICSC50631.2021.00032>
- [KR15] KAVYASRUJANA, D. ; RAO, B. C.: Hierarchical Clustering for Sentence Extraction Using Cosine Similarity Measure. In: SATAPATHY, Suresh C. (Ed.) ; GOVARDHAN, A. (Ed.) ; RAJU, K. S. (Ed.) ; MANDAL, J. K. (Ed.): *Emerging ICT for Bridging the Future - Proceedings of the 49th Annual Convention of the Computer Society of India (CSI) Volume 1*. Cham : Springer International Publishing, 2015, p. 185–191
- [Kuhr22] KUHR, Felix: *Context is the Key: Context-aware Corpus Annotation using Subjective Content Descriptions*, University of Lübeck, Diss., 2022. <https://nbn-resolving.de/urn:nbn:de:gbv:841-2022060962>
- [Kum19] KUMMERFELD, Jonathan K.: SLATE: A Super-Lightweight Annotation Tool for Experts. In: *arXiv preprint arXiv:1907.08236* (2019)
- [KWM19] KUHR, Felix ; WITTEN, Bjarne ; MÖLLER, Ralf: Corpus-Driven Annotation Enrichment. In: *13th IEEE International Conference on Semantic Computing, (ICSC 2019), Newport Beach, CA, USA, January 30*

- *February 1*, (2019), p. 138–141. <http://dx.doi.org/10.1109/ICOSC.2019.8665501>. – DOI 10.1109/ICOSC.2019.8665501. – ISSN 2325–6516
- [LGNB11] LAU, Jey H. ; GRIESER, Karl ; NEWMAN, David ; BALDWIN, Timothy: Automatic Labelling of Topic Models. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA : Association for Computational Linguistics, June 2011, 1536–1545
- [LKM16] LANGE, Mona ; KUHR, Felix ; MÖLLER, Ralf: Using a Deep Understanding of Network Activities for Workflow Mining. In: *KI 2016: Advances in Artificial Intelligence - 39th Annual German Conference on AI, Klagenfurt, Austria, September 26-30* Vol. 9904, Springer, 2016 (Lecture Notes in Computer Science), p. 177–184
- [Llo82] LLOYD, S.: Least squares quantization in PCM. In: *IEEE Transactions on Information Theory* 28 (1982), no. 2, p. 129–137. <http://dx.doi.org/10.1109/TIT.1982.1056489>. – DOI 10.1109/TIT.1982.1056489
- [LPP⁺20] LEWIS, Patrick ; PEREZ, Ethan ; PIKTUS, Aleksandra ; PETRONI, Fabio ; KARPUKHIN, Vladimir ; GOYAL, Naman ; KÜTTLER, Heinrich ; LEWIS, Mike ; YIH, Wen-tau ; ROCKTÄSCHEL, Tim et al.: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In: *Advances in Neural Information Processing Systems* 33 (2020), p. 9459–9474
- [LZ19] LIAO, Xiaofeng ; ZHAO, Zhiming: Unsupervised Approaches for Textual Semantic Annotation, A Survey. In: *ACM Computing Surveys (CSUR)* 52 (2019), no. 4, p. 1–45
- [Mav69] MAVERICK, George V.: Computational Analysis of Present-Day American English. Henry Kučera, W. Nelson Francis. In: *International Journal of American Linguistics* 35 (1969), no. 1, 71–75. <https://doi.org/10.1086/465045>
- [MC17] MORATANCH, N. ; CHITRAKALA, S.: A survey on extractive text summarization. In: *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, IEEE, 2017, p. 1–6
- [MCCD13] MIKOLOV, Tomas ; CHEN, Kai ; CORRADO, Greg ; DEAN, Jeffrey: *Efficient Estimation of Word Representations in Vector Space*. 2013
- [Mil95] MILLER, George A.: WordNet: A lexical database for English. In: *Communications of the ACM* 38 (1995), no. 11, p. 39–41

- [NKAJ59] NEWCOMBE, Howard B. ; KENNEDY, James M. ; AXFORD, SJ ; JAMES, Allison P.: Automatic linkage of vital records. In: *Science* 130 (1959), no. 3381, p. 954–959
- [NN15] NGUYEN, Nguyet ; NGUYEN, Dung: Hidden Markov model for stock selection. In: *Risks* 3 (2015), no. 4, p. 455–473
- [PM+00] PELLEG, Dan ; MOORE, Andrew W. et al.: X-means: Extending k-means with efficient estimation of the number of clusters. In: *Icml* Vol. 1, 2000, p. 727–734
- [Ram03] RAMOS, Juan E.: Using TF-IDF to Determine Word Relevance in Document Queries. (2003)
- [RBH15] RÖDER, Michael ; BOTH, Andreas ; HINNEBURG, Alexander: Exploring the Space of Topic Coherence Measures. In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. New York, NY, USA : Association for Computing Machinery, 2015 (WSDM '15). – ISBN 9781450333177, p. 399–408
- [RHW86] RUMELHART, D. ; HINTON, Geoffrey E. ; WILLIAMS, R. J.: Learning representations by back-propagating errors. In: *Nature* 323 (1986), 533–536. <https://doi.org/10.1038/323533a0>
- [RJ86] RABINER, Lawrence R. ; JUANG, BH: A tutorial on hidden Markov models. In: *IEEE ASSP Magazine* 3 (1986), no. 1, p. 4–16
- [RN18] RADFORD, Alec ; NARASIMHAN, Karthik: Improving Language Understanding by Generative Pre-Training. (2018). <https://api.semanticscholar.org/CorpusID:49313245>
- [RN21] RUSSELL, Stuart ; NORVIG, Peter: *Artificial Intelligence, Global Edition – A Modern Approach*. Pearson Deutschland, 2021. – 36–62 p. <https://elibrary.pearson.de/book/99.150005/9781292401171>. – ISBN 978–1–292–40117–1
- [Ros58] ROSENBLATT, Frank: The perceptron: A probabilistic model for information storage and organization in the brain. In: *Psychological review* 65 (1958), no. 6, p. 386
- [RWC+19] RADFORD, Alec ; WU, Jeff ; CHILD, Rewon ; LUAN, David ; AMODEI, Dario ; SUTSKEVER, Ilya: Language Models are Unsupervised Multitask Learners. (2019). <https://api.semanticscholar.org/CorpusID:160025533>

-
- [RZGSS04] ROSEN-ZVI, Michal ; GRIFFITHS, Thomas ; STEYVERS, Mark ; SMYTH, Padhraic: The Author-Topic Model for Authors and Documents. In: *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence* (2004), p. 487–494. ISBN 0974903906
- [SBDS14] SABOU, Marta ; BONTCHEVA, Kalina ; DERCZYNSKI, Leon ; SCHARL, Arno: Corpus Annotation through Crowdsourcing: Towards Best Practice Guidelines. In: *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)* (2014), 01. http://www.lrec-conf.org/proceedings/lrec2014/pdf/497_Paper.pdf
- [SBK05] SANG, Tian ; BEAME, Paul ; KAUTZ, Henry A.: Performing Bayesian Inference by Weighted Model Counting. In: *AAAI Conference on Artificial Intelligence*, 2005
- [SBM23] SIEVERS, Thomas ; BENDER, Magnus ; MÖLLER, Ralf: Connecting AI Technologies as Online Services to a Humanoid Service Robot. In: *15th International Conference on Computer and Automation Engineering (ICCAE)*, IEEE, 2023, 431-435
- [SGM19] STRUBELL, Emma ; GANESH, Ananya ; MCCALLUM, Andrew: Energy and Policy Considerations for Deep Learning in NLP. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy : Association for Computational Linguistics, July 2019, p. 3645–3650
- [SJ72] SPARCK JONES, Karen: A Statistical Interpretation of Term Specificity and its Application in Retrieval. In: *Journal of Documentation* 28 (1972), no. 1, p. 11–21
- [SK17] SHETTY, Krithi ; KALLIMANI, Jagadish S.: Automatic extractive text summarization using K-means clustering. (2017). <https://doi.org/10.1109/ICEECCOT.2017.8284627>
- [SKSM08] SRIVASTAVA, Abhinav ; KUNDU, Amlan ; SURAL, Shamik ; MAJUMDAR, Arun: Credit card fraud detection using hidden Markov model. In: *IEEE Transactions on dependable and secure computing* 5 (2008), no. 1, p. 37–48
- [SKW07] SUCHANEK, Fabian M. ; KASNECI, Gjergji ; WEIKUM, Gerhard: YAGO: A core of semantic knowledge. In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, 2007, p. 697–706

- [SM23] SCHIFF, Simon ; MÖLLER, Ralf: Persistent Data, Sustainable Information. In: *Proceedings of the Workshop on Humanities-Centred Artificial Intelligence (CHAI 2023)*, CEUR Workshop Proceedings, 2023, 5–14
- [SN08] SENO, Eloize Rossi M. ; NUNES, Maria das Graças Volpe: Some Experiments on Clustering Similar Sentences of Texts in Portuguese. In: TEIXEIRA, António (Ed.) ; LIMA, Vera Lúcia S. (Ed.) ; OLIVEIRA, Luís C. (Ed.) ; QUARESMA, Paulo (Ed.): *Computational Processing of the Portuguese Language*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, p. 133–142
- [TRH16] TOWNE, W. B. ; ROSÉ, Carolyn P. ; HERBSLEB, James D.: Measuring Similarity Similarly: LDA and Human Perception. In: *ACM Trans. Intell. Syst. Technol.* 8 (2016), no. 1, 7:1–7:28. <https://doi.org/10.1145/2890510>
- [VIN15] VIJAYARANI, S. ; ILAMATHI, J. ; NITHYA, S.: Preprocessing Techniques for Text Mining - An Overview. In: *International Journal of Computer Science & Communication Networks* 5 (2015), p. 7–16
- [Vit67] VITERBI, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In: *IEEE Transactions on Information Theory* 13 (1967), no. 2, 260–269. <https://doi.org/10.1109/TIT.1967.1054010>
- [VSP⁺17] VASWANI, Ashish ; SHAZEER, Noam ; PARMAR, Niki ; USZKOREIT, Jakob ; JONES, Llion ; GOMEZ, Aidan N. ; KAISER, Lukasz ; POLOSUKHIN, Illia: Attention Is All You Need. (2017). <https://arxiv.org/abs/1706.03762>
- [YCZS14] YANG, Libin ; CAI, Xiaoyan ; ZHANG, Yang ; SHI, Peng: Enhancing sentence-level clustering with ranking-based clustering framework for theme-based summarization. In: *Information Sciences* 260 (2014), 37–50. <https://doi.org/10.1016/j.ins.2013.11.026>. – ISSN 0020–0255
- [YYH23] YANG, Hui ; YUE, Sifu ; HE, Yunzhong: *Auto-GPT for Online Decision Making: Benchmarks and Additional Opinions*. <https://arxiv.org/pdf/2306.02224.pdf>. Version: 2023
- [YZG⁺20] YANG, Ziyi ; ZHU, Chenguang ; GMYR, Robert ; ZENG, Michael ; HUANG, Xuedong ; DARVE, Eric: TED: A Pretrained Unsupervised Summarization Model with Theme Modeling and Denoising. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online : Association for Computational Linguistics, November 2020, p. 1865–1874

- [YZLL18] YANG, Jie ; ZHANG, Yue ; LI, Linwei ; LI, Xingxuan: YEDDA: A Lightweight Collaborative Text Span Annotation Tool. In: *Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, System Demonstrations*, 2018, p. 31–36
- [Zha04] ZHANG, Yingjian: *Prediction of financial time series with Hidden Markov Models*, Applied Sciences: School of Computing Science, Diss., 2004
- [ZLWZ18] ZHANG, Xingxing ; LAPATA, Mirella ; WEI, Furu ; ZHOU, Ming: Neural Latent Extractive Document Summarization. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium : Association for Computational Linguistics, October-November 2018, p. 779–784

Publications of Magnus Bender

2024

- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke **Enhancement of Subjective Content Descriptions by using Human Feedback** to be published in *International Journal of Semantic Computing, 2024* <https://arxiv.org/abs/2405.15786>
- Magnus Bender **Automate Text Processing for Schematically Analyzing Legal Texts** to be published in *Proceedings of the Humanities-Centred AI (CHAI) Workshop at KI2024, 47th German Conference on Artificial Intelligence, 2024*
(Slides: <https://dx.doi.org/10.25592/uhhfdm.16138>)
- Hagen Peukert, Lucas F. Voges, Thomas Asselborn, Magnus Bender, Ralf Möller, Sylvia Melzer **Humanities in the Center of Data Usability: Data Visualization in Institutional Research Repositories** to be published in *Proceedings of the Humanities-Centred AI (CHAI) Workshop at KI2024, 47th German Conference on Artificial Intelligence, 2024*
(Slides: <https://dx.doi.org/10.25592/uhhfdm.16138>)
- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **ReFrESH – Relation-preserving Feedback-reliant Enhancement of Subjective Content Descriptions** in *18th IEEE International Conference on Semantic Computing (ICSC 2024)* – Best Paper Award
<https://dx.doi.org/10.1109/ICSC59802.2024.00010>
- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **Unsupervised Estimation of Subjective Content Descriptions in an Information System** in *International Journal of Semantic Computing, 2024*
<https://dx.doi.org/10.1142/S1793351X24410034>

2023

- Magnus Bender, Kira Schwandt, Ralf Möller, Marcel Gehrke: **FrESH – Feedback-reliant Enhancement of Subjective Content Descriptions by Humans** in *Proceedings of the Humanities-Centred AI (CHAI) Workshop at KI2023, 46th German Conference on Artificial Intelligence, 2023*
<https://ceur-ws.org/Vol-3580/paper3.pdf> (Slides: <https://dx.doi.org/10.25592/uhhfdm.13423>)
- Thomas Asselborn, Sylvia Melzer, Said Aljoumani, Magnus Bender, Florian Andreas Marwitz, Konrad Hirschler and Ralf Möller: **Fine-tuning BERT Models on Demand for Information Systems Explained Using Training Data from Pre-modern Arabic** in *Proceedings of the Humanities-Centred AI (CHAI) Workshop at KI2023, 46th German Conference on Artificial Intelligence, 2023*
<https://ceur-ws.org/Vol-3580/paper5.pdf> (Slides: <https://dx.doi.org/10.25592/uhhfdm.13423>)
- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **LESS is More: LEan Computing for Selective Summaries** in *KI 2023: Advances in Artificial Intelligence. Lecture Notes in Computer Science, Springer*
https://dx.doi.org/10.1007/978-3-031-42608-7_1
- Thomas Sievers, Magnus Bender, Ralf Möller: **Connecting AI Technologies as Online Services to a Humanoid Service Robot** in *15th International Conference on Computer and Automation Engineering (IC-CAE 2023)*
<http://dx.doi.org/10.1109/ICCAE56788.2023.10111181>
- Magnus Bender, Tanya Braun, Ralf Möller, Marcel Gehrke: **Unsupervised Estimation of Subjective Content Descriptions** in *17th IEEE International Conference on Semantic Computing (ICSC 2023)*
<https://dx.doi.org/10.1109/ICSC56153.2023.00052>

2022

- Simon Schiff, Magnus Bender, Ralf Möller: **Embodiment of an Agent by a Pepper Robot for Explaining Retrieval Results** in *Proceedings of the Humanities-Centred AI (CHAI) Workshop at KI2022, 45th German Conference on Artificial Intelligence, 2022*
<https://ceur-ws.org/Vol-3301/paper4.pdf> (Slides: <https://dx.doi.org/10.25592/uhhfdm.10769>)

- Magnus Bender, Felix Kuhr, Tanya Braun: **To Extend or not to Extend? Enriching a Corpus with Complementary and Related Documents** in *International Journal of Semantic Computing*, 2022
<https://dx.doi.org/10.1142/S1793351X2240013X>
- Magnus Bender, Felix Kuhr, Tanya Braun: **To Extend or not to Extend? Complementary Documents** in *16th IEEE International Conference on Semantic Computing (ICSC 2022)*
<https://dx.doi.org/10.1109/ICSC52841.2022.00011>
- Magnus Bender, Felix Kuhr, Tanya Braun, Ralf Möller: **Estimating Context-Specific Subjective Content Descriptions using BERT** in *16th IEEE International Conference on Semantic Computing (ICSC 2022)*
<https://dx.doi.org/10.1109/ICSC52841.2022.00034>

2021

- Magnus Bender, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, Simon Schiff: **Identifying and Translating Subjective Content Descriptions Among Texts** in *International Journal of Semantic Computing*, 2021
<https://dx.doi.org/10.1142/S1793351X21400122>
- Felix Kuhr, Magnus Bender, Tanya Braun, Ralf Möller: **Context-specific Adaptation of Subjective Content Descriptions** in *15th IEEE International Conference on Semantic Computing, (ICSC 2021)*
<https://dx.doi.org/10.1109/ICSC50631.2021.00032>
- Magnus Bender, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, Simon Schiff: **Identifying Subjective Content Descriptions Among Texts** in *15th IEEE International Conference on Semantic Computing (ICSC 2021)*
<https://dx.doi.org/10.1109/ICSC50631.2021.00008>

2020

- Felix Kuhr, Magnus Bender, Tanya Braun, Ralf Möller: **Augmenting and Automating Corpus Enrichment** in *International Journal of Semantic Computing*, 2020
<https://dx.doi.org/10.1142/S1793351X20400061>
- Felix Kuhr, Magnus Bender, Tanya Braun, Ralf Möller: **Maintaining Topic Models for Growing Corpora** in *14th IEEE International Conference on Semantic Computing (ICSC 2020)*
<https://dx.doi.org/10.1109/ICSC.2020.00087>

2019

- Felix Kuhr, Tanya Braun, Magnus Bender, Ralf Möller: **To Extend or not to Extend? Context-specific Corpus Enrichment** in *Proceedings of AI 2019: Advances in Artificial Intelligence, 2019, Springer*
https://dx.doi.org/10.1007/978-3-030-35288-2_29

Curriculum Vitae

Personal Information

Magnus Bender



Professional Experience

Since April 2024	Research Associate at the Institute of Humanities-Centered Artificial Intelligence (CHAI), Universität Hamburg
March 2021 to March 2024	Research Assistant at the Institute of Information Systems, University of Lübeck

Education

September 2021	M.Sc. in Computer Science, University of Lübeck
October 2019	B.Sc. in Computer Science, University of Lübeck
July 2016	General Qualification for University Entrance (Abitur), Ernestinenschule Lübeck

On the Web

DBLP: <https://dblp.org/pid/253/6981.html>

Scholar: https://scholar.google.com/citations?user=A_HVBuMAAAAJ

ORCID: <https://orcid.org/0000-0002-1854-225X>