Universität Osnabrück
Fachbereich Humanwissenschaften
Institute of Cognitive Science

Master's thesis

Data-Driven Embedding of Educational Resources in a Vector Space with Interpretable Dimensions for Explainable Recommendation

Christoph Stenkamp
955004
Master's Program Cognitive Science
April 2017 - April 2022

First supervisor: Dr. Tobias Thelen
Institute of Cognitive Science
University of Osnabrück

Second supervisor: Johannes Schrumpf, M.Sc.
Institute of Cognitive Science
University of Osnabrück

Declaration of Authorship

I hereby certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

_____ _____

city, date signature

Abstract

The number of available educational resources keeps rising just as the liberty for students to create a custom curriculum, making recommendation in this sector ever more critical. There is the need for a methodology to structure these resources unsupervisedly to allow for explainable recommendations, where users can explicitly demand features such as the degree how mathematical a course should be. Conceptual Spaces allow to embed resources into a vector space whose quality dimensions correspond to salient semantic features, but generating them reliably is an open question. This thesis replicates a method to create these spaces data-driven from their descriptions and applies it to a given dataset of that domain to explore its applicability. Our preliminary results show that human categories such as the faculty a course belongs to are among the detected salient directions. Further qualitative analysis also indicates that the embeddings capture essential features that can be used in interpretable classification and explainable recommendation. Additionally, a scalable and modular architecture for the algorithm is created. It is shown that it can be applied to diverse variations of dataset and algorithm components. A workflow is implemented and described to run the algorithm efficiently on compute clusters. The full implementation is open-sourced to make future research on related problems more accessible.

Contents

List of Figures

List of Tables

1. Introduction

In this thesis, we want to generate a conceptual space for the domain of university courses automatically in a data-driven way from their descriptions.

1.1. Reading Instructions

Document Structure After these reading instructions, **chapter one** continues the Introduction by stating why the topic of this thesis is important before defining the exact goals that shall be achieved. The **second chapter** then discusses some relevant background. This includes information about the SIDDATA-project, aspects to consider for software replication and quality, the idea of conceptual spaces in general, and a short introduction to the base algorithm as well as some important concepts and techniques required for it. The **Methods** then goes into detail about the used dataset as well as the implemented algorithm and architecture, followed by information on the workflow used to generate results and the metrics to analyse them. The **fourth chapter** then reports the generated results, first comparing those for a dataset from the literature to demonstrate technical correctness, followed by quantitative and qualitative results as well as optimal hyperparameters for the main dataset. These are interpreted and discussed in the **final chapter** before they are put into the context of the aims of this thesis. This is followed by future research directions before everything is wrapped up in the Conclusion.

Digital Version This thesis is compiled both as a two-page printable version and as a highly hyperlinked version recommended when reading the document digitally. In the latter, all links are colored, all glossary entries and abbreviations are linked, and it is possible to jump forward and back for sections, citations and references. After jumping to a referenced figure from elsewhere, it is possible to jump back (on supported Readers) by clicking the arrow in its caption. In cases where e. g. evaluation metrics are established in another chapter than where they are used, additional hyperlinks allow jumping forward and backward. Both versions of this thesis can be downloaded from its repository.[1]

Unfortunately, both this thesis's printed and PDF versions do not allow for interactive elements, making three-dimensional plots hard to understand. Because of that, where appropriate only 2D versions are printed together with a reference to an online 3D version. It is highly recommended to follow these links, as they contain interactive plots that can be twisted and turned to truly explore their dimensions and show much additional information that cannot be statically shown, such as detailed information about the respective entities of a scatterplot on mouse-over.

Regarding Terminology This thesis uses the "we" as *Pluralis Auctoris*, signifying objectivity in science. This does not mean that anyone but the author of this thesis contributed to it unless explicitly stated otherwise. Many abbreviations, symbols, and technical

[1] `https://github.com/cstenkamp/MastersThesisText/`. Direct link to compiled documents: `https://nightly.link/cstenkamp/MastersThesisText/workflows/create_pdf_artifact/master/Thesis.zip`

terms will be used throughout the text. Those that cannot be expected to be known by the reader will be clearly defined - either in the section about Relevant Algorithms and Techniques or the Conclusion at the end of this thesis. The glossary discriminates between clearly defined *Definitions* and *Custom Terms*, where concepts often referred to in this text are comprehensively explained without any claim to their true definition.

1.2. Motivation

1.2.1. Course Recommendation

This thesis explores a novel method to generate explainable recommendations for university courses and other educational resources. The results will eventually be incorporated into the *Siddata*-platform, helping students to find courses that fit their interests.

Overwhelming amounts of resources

The university landscape has changed drastically in recent years. In 1999, the *Bologna declaration* was signed by the 29 (now 45) countries of the *European Higher Education Area*, reforming their education systems to allow for international compatibility. While before 1999 there were between 70 and 180 different elementary studies in Germany [4], that number had risen to 2554 in 2003,[2] had become more than 5000 in 2008 and currently (winter semester 2020/2021) peaks at 9168 unique courses of study.[3] The number of subjects in total has almost doubled in the past thirteen years, from 11 265 in 2007 to 20 359 in 2020.[3] According to the German *Centrum für Hochschulentwicklung*, nowadays only 18.7% of degrees are *classical*, i. e. tailored to a specific subject such as *Physics*, the rest fall unter the categories hybrid, interdisciplinary or topic-focused.[4]

Where before there was a rigid schedule of mandatory courses, modern courses of studies are becoming increasingly modular, allowing students to draw up individual educational plans composed of a wide selection of courses.[5] Due to globalisation and the Introduction of the *European Credit Transfer System*, the selection of courses for this may span any course at any European university.

Finally, thanks to increasing digitalization and especially boosted in recent years by the COVID-19-Pandemic, the number of publicly available OERs has skyrocketed. For example, the number of MOOCs available on the e-learning platform *Udemy*[6] has increased from 20 000 in 2015 to more than 157 000 in January of 2021.[7]

Academics nowadays must engage with a multitude of interconnected, digital and open practices and technologies [5]. High-quality OERs become more and more widespread

[2] `https://www.che.de/download/im_blickpunkt_ausdifferenzierung_studiengaenge-pdf/?wpdmdl=10620&refresh=624af74f6f7921649080143` (accessed at 14th March 2022)

[3] `https://www.hrk.de/fileadmin/redaktion/hrk/02-Dokumente/02-03-Studium/02-03-01-Studium-Studienreform/HRK_Statistik_BA_MA_UEbrige_WiSe_2020_21_finale.pdf`, page 10 (accessed at 14th March 2022)

[4] `https://www.che.de/wp-content/uploads/upload/Im_Blickpunkt_Die_Vielfalt_der_Studiengaenge_2017.pdf` (accessed at 14th March 2022)

[5] `https://www.pedocs.de/volltexte/2008/285/pdf/heft98.pdf` (accessed at 14th March 2022)

[6] `https://www.udemy.com/`

[7] `https://www.classcentral.com/report/udemy-by-the-numbers/` (accessed at 15th March 2022)

and "may ultimately be the genuine equalizer for education and for empowering social inclusion in a pluralistic, multicultural, and imperfect world" [6, p. 2]. All these trends fundamentally change the landscape of higher education, leaving students with overwhelming quantities of high-quality educational resources available. As, however, the time at the hand of the students is now as limited as before, the choice of the right resources in this ocean of information becomes increasingly problematic. Locating, retrieving and differentiating available resources becomes more and more challenging [5].

The *Future Skills Report*[8] on the future of learning and higher education [7] suggests that this trend will continue: According to the study, future academic education will look fundamentally different from today, in that it will likely become increasingly multi-institutional with students individually having their own personalized, flexible curriculum selected from a vast set of resources, compared to which the currently available study programmes are as rigid as they have ever been [7].

Explainable Recommendation

The Siddata-platform already contains a tool that generates recommendations based on academic interests specified by the user [8]. The novel idea explored here, however, shall fall under the realm of **Explainable Artificial Intelligence** and work interactively, with the user *in the loop*. The use case considered for this thesis is the following: A system should be found that provides well-founded recommendations for resources, based on input and feedback by the user. A specific sample interaction that shall be made possible[9] is a user requesting

»A course like Codierungstheorie und Kryptographie*, but with less maths.«*

To be able to work with such requests, the system would need some sort of *feature directions*: It must recognize *math* as a feature that any course may have and it must be able to rank all courses according to how much the feature *math* applies to each of them.

For more than twenty years, the default approach for recommendation has been that of *Collaborative Filtering*, summarised in the well-known phrase "*Customers who bought items in your Shopping cart also bought: ...*" [9]. Traditionally, this algorithm represents every customer as a vector, whose components are the number of times this customer has bought an item for all items of the store. To suggest new products to the user, items from the vectors of customers whose purchase history is similar to this user, calculated, for example, by the cosine distance, are suggested [10]. Because this is computationally very expensive, there are several alternatives, for example, algorithms that cluster users based on their similarity to other users before matching, or search-based algorithms. Even Amazon's recommendation system, one of the most world-defining algorithms in the world, uses the same base algorithm, coined "*item-to-item collaborative filtering*". It only differs from the traditional techniques in that it builds a similarity matrix based on the cosine distance for items instead of customers. Developed in 1997, this algorithm is not only still in use at Amazon today, but has also been adopted by Youtube and Netflix, amongst many others [11]. Many improvements in efficiency, distance calculation

[8] `www.nextskills.org`

[9] ... and also happens to be the exact request the author of this thesis had some time ago

techniques or time-dependency have been added since then, and the algorithm is likely more optimized than most others. Yet, the fact that it is a simple *similarity-based reasoning technique* has remained ever since. As will be explained in more detail in Section 2.3.2, both in terms of classification algorithms and classic logical reasoning, the technique to classify a sample with the class of its most similar neighbors (*k-Nearest-Neighbors*) is considered one of the most basic techniques.

On a different note, *feature directions* are not unknown in the field of Computational Linguistics. As famously demonstrated by Mikolov *et al.* [12] in their 2013 paper "Linguistic Regularities in Continuous Space Word Representations", modern neural language models that represent words as high-dimensional continuous-space vectors exhibit astonishing semantic regularities. Not only does the usage of such embeddings boost the performance of many classical NLP tasks [13–15], but there is also strong evidence that VSMs capture the meaning of words on the basis of the distributional hypothesis. This best shows in the famous example that links simple vector arithmetic to word semantics:[10]

$$\begin{aligned} vec(king) - vec(man) + vec(woman) &\approx vec(queen) \\ vec(planet) + vec(water) &\approx vec(earth) \\ vec(house) + vec(movie) &\approx vec(cinema) \end{aligned} \tag{1.1}$$

This can be considered semantic directions and would seem to allow the example stated at the start of this section. However Word2Vec is not suited to allow data-driven explainable recommendation as demanded for the use-case of this thesis. This is because in these embeddings, there are no meaningful **unit vectors**. There is no obvious direction designating a gender in this space, but instead *man* is one of millions of vectors in this space, its direction obfuscated throughout all of its vector components, and the actual direction solely depends on the initial random distribution.

Intuitively, a vector-space that allows for the kind of explainable recommendation we are looking for would need to be specified such that there are a few most basic *bare properties* of all entities of the respective domain. These properties would correspond to linearly independent unit vectors, which are the *dimensions* spanning the vector space: **human-interpretable feature directions**.

This would allow to rank objects according to how much they apply to each of a few basic features, allowing a wide variety of tasks, such as interpretable rule-based classifiers, search engines working with gradual and ill-defined features (e. g. *popular movies*), or critique-based recommender systems with the user in the loop [2]. Most importantly, such a ranking could be used to build up a *structured knowledge base* for the domain.

In our search of literature with related goals, the approach of [16] stood out: In their paper, they create a structured knowledge base for the domain of movies and provide an interface that allows a user to select a movie based on a set of clear-defined semantic features. This interface is reprinted in Figure 1.1.

While the result of their algorithm aligns very much with the desired goal of this work, their implementation relies on supervised learning that requires a hand-labelled dataset.

[10] Latter two examples adapted from `https://devmount.github.io/GermanWordEmbeddings/` (accessed at 14th March 2022)

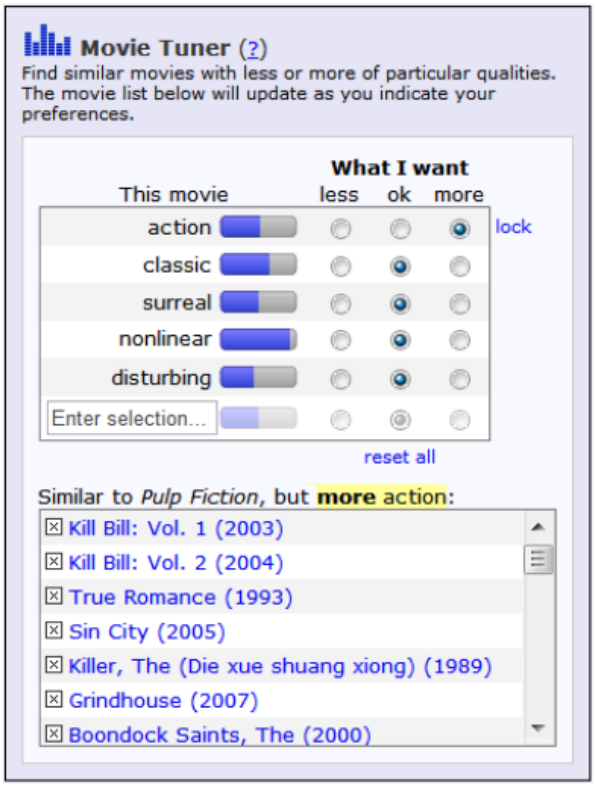

Figure 1.1.: The Movie Tuner Interface from Vig *et al.* [16]. Figure reprinted from [17, Fig.3] (permission for non-commercial use granted).

1.2.2. The algorithm of Derrac & Schockaert [1]

To summarise, the kind of recommendation aimed for here would require a structured knowledge base that can be created from any domain in a data-driven way. This knowledge base can be represented as a vector space of human-interpretable, linearly independent components. A Euclidean distance metric for the space would ease human interpretation.

Famously, the idea of **Conceptual Spaces** fulfills these criteria. Introduced by Peter Gärdenfos in his 2000 book *Conceptual Spaces: The Geometry of Thought* [18] as a bridge between symbolic and subsymbolic processing, conceptual spaces represent knowledge in a geometric structure consisting of various quality dimensions. While well-known primarily as a theoretical model, an algorithm to automatically generate such spaces in a data-driven way was proposed by Derrac & Schockaert [1] in 2015.

Their primary motivation is the following: Classical symbolistic AI relying on semantic knowledge bases can provide intuitive explanations for its decisions, but automatic creation of such has thus far been unsuccessful. Inspired by the way humans deal with incomplete knowledge, they draw a connection between commonsense reasoning patterns and the relation of concepts and entities in a conceptual space, claiming that qualitative spatial relations in the latter correspond to the semantic relations required for reasoning that help to fill gaps in these knowledge bases. In doing so, they create multiple classifiers that provide intuitive explanations for each decision.

Given the widely accepted assumption that, algorithmically, recommendation is nothing beyond classification [9, 10, 19], Derrac & Schockaert's [1] algorithm seems the perfect base to allow for explainable recommendation as required here if it works reasonably well for the given domain.

Precise study of the pertinent literature has shown that there is a small but active community publishing improvements to this algorithm. However, most of these follow-up works include one of the authors of the primary paper, Steven Schockaert, as co-author [2, 3], indicating a small impact beyond this community. Furthermore, most of the considered literature use manually created datasets collected from multiple human reviews or tags, and given that the algorithm relies on the fact that relevant words occur more often in the text corpus, there is reason to assume that it may struggle when applied to the corpus given in this thesis. For these reasons, it seemed reasonable to thoroughly test the domain transfer of the algorithm.

Unfortunately, the available open-source implementations that were found[11] proved undocumented and obscure,[12] which is why it was decided that a complete re-implementation is the better idea and also less work. This also resulted in a new goal, namely, to create a reliable architecture for the algorithm that adheres to modern standards of software quality, as specified in **ISO/IEC 9126** and its replacement **ISO/IEC 25010:2011**, hopint to make future research just like this one - where the validity of the algorithm and its ability to transfer to new domains is tested - easier than it was for the author of this thesis.

1.3. Research Questions and Thesis Goals

In summary, the primary motivation for this thesis is to test if the methodology of Derrac & Schockaert [1] is applicable to the domain of educational resources to find regularities in the data and allow for explainable recommendation of these.

The methodology relies on many modular components with no obvious reason to choose any hyperparameter or component over another, resulting in a combinatorial explosion of runs, each with a significant runtime. Throughout the development of the respective code, the importance of a solid software-foundation became clear, leading to a shift in focus away from quickly generating results towards the methodology for a scalable, adaptable and qualitative architecture that may serve as foundation to answer many related research questions in the future. Among others, this includes the application of the algorithm to compute clusters such as the IKW Grid Engine. The domain transfer to educational resources shall, however, still serve as a prototypical application for which answers will be generated.

So, this thesis shall not only replicate the results of [1] and transfer their algorithm to a domain with practical benefits, but also deliver qualitative software for future research.

[11] [3]: `https://github.com/rana-alshaikh/Hierarchical_Linear_Disentanglement`
[2]: `https://github.com/ThomasAger/Autoencoder-Explanations` (accessed at 24th January 2022)

[12] Relying for example on 70 unnamed command-line arguments: `https://github.com/ThomasAger/Autoencoder-Explanations/blob/master/src/_archive/lr_pipeline.py#L1211-L1283` (accessed at 24th January 2022)

To make this even easier, there will be a strong textual focus on the architecture as well, in the hope of allowing other scientists to work with this code-base. To show that the implementation works, it will be also applied to an original dataset of [1] and the results will be compared.

Clearly defined, the two stated goals of this thesis are thus:

1. Implement and describe a qualitative, scalable and adaptable software architecture for the replicated algorithm that can be used to easily apply it to new domains.
2. Apply the methodology to the domain of educational resources to verify if the algorithm works and is useful to find regularities in it, allowing for explainable recommendation. In terms of original contributions, this includes finding differences in the structure of the given datasets and, where appropriate, providing additions to the algorithm such that it works not only on specifically curated datasets.

1.3.1. Success conditions

Exact evaluation metrics will be explained in detail later, but let us quickly define a few conditions that indicate successful execution of these goals.

Regarding the architecture

A successful architecture is given if

- It successfully runs on the compute grid, indicating scalability and modularity.
- The results of [1] and its follow-up works [2, 3] can be replicated for at least one of their original datasets.
- The criteria for software quality as specified in **ISO 25010** are fulfilled.
- The code is open-sourced, well-documented and understandable.

Regarding the application of the algorithm to a new domain

The main question to be answered here is *Is the algorithm applicable for the domain of educational resources*. This question must be explicitly answered by first stating the hypothesis, explaining the methods used, reporting results and subsequently discussing them. Unfortunately, the main success criterion of the algorithm is its subjective persuasiveness, which is hard to objectively and quantitatively evaluate without a study. For that reason, there is the need to find good surrogate metrics instead. Criteria for success include:

- The difference from the given dataset to the originally used ones is elaborated and regarded for.
- Proper scientific methodology will be conducted for the evaluation of the results.
- The resulting semantic directions are *convincing* in a sense that they fulfill demanded criteria that were specified before looking at results.
- The resulting semantic directions can be used as the basis for explainable recommenders.
- These recommenders achieve similar performances to modern ML techniques when applied to find human categories such as a course's faculty from the data.

2. Background

This chapter will introduce all necessary concepts relevant for the methodology implemented in this work. For that, it will explain some important concepts for software and replication and introduce the SIDDATA-project that revolves around the domain educational resources. After that, Conceptual Spaces, the algorithm to generate them unsupervisedly and how reasoning using them can be performed is introduced. Finally, the work is put into the context of computational NLP with an overview of related research as well as the theoretical foundation of the algorithm and its components.

2.1. Replication and Software Quality

Having established the goals of replicating an algorithm for a new domain, let us look at how such a replication should be performed and how software quality can be measured.

2.1.1. Replication and Reproducibility

The workflow of data science generally follows the same pattern: A paper states there is some problem X, claims that their algorithm Y may be good at problem X, creates datasets Z for X, and then tests the code on these datasets. This test generally compares X to alternative approaches from the literature and explores if any regularities in the algorithm Y can be found. This may yield future research opportunities, showing what other domains the algorithm may work for as well.

Replication fills this role by applying an existing algorithm to another domain. Results for this are important, as it helps to see, first, if the claimed results are valid and if they work on datasets that are not artificial and specifically created for the sole purpose of testing the algorithm. Furthermore, the details of experiments in published work are often opaque and omit important information to reproduce the algorithm. These issues are mitigated trough repetition: The robustness of the algorithm to changes in parameters or datasets is investigated. If changes in either of these have a major impact on the results, there is reason to doubt the generalization of the algorithm, showing that it may not be good to solve problem X after all.

It is absolutely crucial in science to ensure that all claims that are made are reproducible and testable, ensuring ease of replication. Reproducibility is the pinnacle of *Open Science*.[1] And "Open Science is just science done right".[2] Being a hot topic in psychology since the reproducibility crisis,[3] the topic is just es relevant in computer science

[1] There is no single definition of open science. However reproducibility appears in most tries, such as e. g. `https://www.talyarkoni.org/blog/2019/07/13/i-hate-open-science/` (accessed at 25th March 2022)

[2] Quote from John Tennant, see e. g. `https://soundcloud.com/tidningen-curie/jon-tennant-open-science-is-just-science-done-right` (accessed at 25th March 2022).

[3] Baker M: 1,500 scientists lift the lid on reproducibility. Nature. 2016; 533(7604): 452-4. `https://www.nature.com/articles/533452a` (accessed at 25th March 2022)

research.[4] In that realm, Reproducibility may be seen as sub-goal of (the more fundamental) Sustainability, as e. g. by Mölder *et al.* [20], who claim that "reproducibility alone is not enough to sustain the hours of work that scientists invest in crafting data analyses". To ensure that the analysis performed in this thesis is sustainable and adheres to best scientific and software quality standards, let us find ways to formally define them.

2.1.2. Software Quality

The International Organization for Standardization (ISO) provides an official international standard for the evaluation of software quality as **ISO/IEC 25010:2011** [21]. The full title of the norm is *ISO/IEC 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE)*. It has the objective to ensure the quality of software by providing objective and clearly defined standards for definitions of success. It classifies software quality in eight characteristics, which each consists of several sub-characteristics. The main characteristics are *Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability* and *Portability*. Only some of these goals are relevant to projects like this. However, many subgoals align with the hierarchy of aspects to consider for sustainable data analysis as published by Mölder *et al.* [20], which is is reprinted as Figure 2.1.

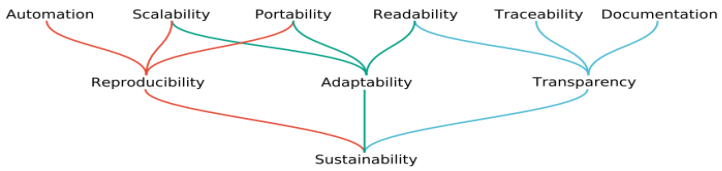

Figure 2.1.: Hierarchy of aspects to consider for sustainable data analysis. Reproduced from [20, Fig.1] (Creative Commons Attribution License)

Important aspects to conduct proper computer science and data analysis that allows for **Sustainability** - allowing the analysis to be of lasting impact - thus include [20, 21]:

Functional Suitability, which means complete, correct and appropriate functionality.

Reproducibility, i. e. allowing validation and regeneration of results on the original or new data. Entails understandable and well documented code as well as *Stability*.

Maintainability and Adaptability, i. e. the ability to modify the analysis to answer extended or slightly different research questions by allowing modifications.

Transparency, i. e. the ability for others to understand it well enough to judge if it's technically as well as methodologically valid - also ensuring Understandability, Appropriateness and Accessibility, Analyzability and Testability.

Scalability, i. e. enabling the scalable execution of the algorithm and each involved step, including deployment on complex compute clusters, grids or clouds. This includes Performance Efficiency and efficient Resource Utilization.

Modularity, i. e. changes in one component have minimal impact on others, allowing for easy exchange and extension. Also entails *Changeability*.

[4] Mesirov JP: Computer science. Accessible reproducible research. Science. 2010; 327(5964): 415-6. `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3878063/` (accessed at 25th March 2022)

2.2. SIDDATA and Educational Resources

To get a better understanding of the domain, this section elaborates on the specific use case of recommendation of educational resources that shall be handled, and introduces the SIDDATA project and platform under which this thesis was developed.[5]

Educational resources are not clearly defined. The term may refer to transcribed videos, scientific papers or books, lecture material such as slides or notes, or even multimedia-data such as MOOCs. In the context handled here, it generally refers to (descriptions of) university courses. This thesis was started while working at the SIDDATA project, with the idea to add a recommender to the platform that can generate course recommendations with the user *in the loop*. SIDDATA is a joint interdisciplinary project for "*Individualization of Studies through Digital, Data-Driven Assistants*"[6] of the universities Osnabrück, Bremen and Leibniz Universität Hannover, funded by the German *Federal Ministry of Education and Research*.[7]

The project adresses the same problem as stated in the Introduction (1.2.1), namely that e-learning and the amount of avilable resources increase, making the choice of right resources an increasingly relevant problem for the learning success of students. Its deliverable is a flexible data-driven DSA, that supports students in higher education in their invidual learning and achievement of personal study goals by giving hints, reminders and recommendation for their individual study paths [8], helping students in setting and achiving individual and self-regulated personal educational goals. This is in line with the increasing importance of skills such as self-organized knowledge acquisition and self-regulatory competencies due to increasing importance of individualization in educational paths of the globalized learning environment [7, 8].

For that, the collaborative project combines heterogenous data and information in a digital study assistant. Data is collected from multiple sources, such as the LMS, offers and resources of other universities and institutions, and data collected from its users. To allow for this heterogeniety and also future extensions, for example new data sources such as MOOCs through several apis APIs, different front-ends, or different recommendation methods, the system relies on a highly modular and extensible architecture. The Frontend is realized as a plugin for the the university's LMS Stud.IP [22]. This not only allows for easy user access, but also to get data about courses from the LMS using cronjobs. The Frontend is connected over a RESTful API to the Backend, which is written in Python on basis of the Web Framework *Django* and relies on a relational *PostgreSQL* database to store information.

The Backend consists of seperate encapsulated recommender modules in a loosely coupled architecture and a common ontology, allowing to easily add new subsystems. The modules generate recommendations towards personal educational goals on basis of the collected data, which are displayed to the user in the Frontend. What comprises a recommender is grouped from a user perspective, such that each recommender focuses on a topic. The

[5] As Siddata signifies both the project and the developed digital assistant, the all-upper 'SIDDATA' henceforth refers to the project, while the specific developed software will be denoted 'Siddata' or 'DSA'.

[6] `https://www.siddata.de/en/`

[7] BMBF. Funding number: 16DHB2124

currently implemented recommenders include, for example, one to find peers with similar interests, get information about scientific careers, personality-based learning behaviour- and study tips, or information regarding local and remote courses and OERs. Another module recommends courses using a combination of rule-based and modern ML techniques that relate natural language queries with the courses known to the system (picked up in Subsection 2.4) [8].

The system is currently in its third prototype, and preliminary evaluation has shown that modules that provide personal recommendation are most well received [8]. This and the ease of use to add new recommenders indicate a high likelyhood of success for adding a new module that recommends courses in the way described above.

The dataset used here was collected through the Siddata platform, which collected courses and events from the three universities currently connected to it, as well as other sources for MOOCs and other OERs through respective APIs (more details in Section 3.1.1). It should be noted that the dataset is not artificially generated (unlike [1–3]) but collected from current courses and their descriptions - making an algorithm for this domain incorporated as recommender to the platform a contribution with practical application.

2.3. Conceptual Spaces

This section will introduce CSs as tool of choice and introduces the replicated algorithm to generate them unsupervisedly. Standing in between classical symbolic artificial intelligence and modern black-box techniques, CS can serve as a bridge that allow to model commonsense reasoning geometrically. Some examples for that will be elucidated later in this section.

Theory of Conceptual Spaces

The theory of Conceptual Spaces was first introduced by Peter Gärdenfors in his 2000 book *Conceptual Spaces: The Geometry of Thought* [18, 23] both as a theoretical model of human concept formation and a format for knowledge representation in artificial systems.

The recent years have seen a successively increase of the dichotomy between **symbolistic** and **connectionistic** approaches to knowledge representation. The former explicitly model knowledge representation and reasoning through e. g. logical calculi, sets of rules or semantic knowledge bases. This allows for explicit and explainable re-creation of high-level, abstract human reasoning such as deduction (*syllogisms*), but requires explicit manual creation of abstract knowledge bases such as ontologies, lexicons and sets of rules. The other side consists of modern ML techniques such as ANNs and other data science algorithms. Many of these techniques can work with noisy, high-dimensional data, such as direct visual input. On real world datasets, they generally outperform classical algorithms with high margins whilst only requiring some *target* label. However, they are *black boxes*, meaning that their inner workings cannot be inspected or manually tweaked.[8]. CSs can serve as a bridge between symbolistic and connectionistic approaches

[8] And also blindly replicate biases in the data, see e. g. `https://hbr.org/2019/10/what-do-we-do-about-the-biases-in-ai` (accessed at 5th April 2022)

to knowledge representation. By having CSs as a layer of reasoning and representation in between both, classical knowledge base *symbols* would be grounded in noisy high-dimensional data, allowing for high-level syllogistic reasoning from real-world data.

According to Gärdenfors, concept-representation in humans is represented by three levels of accounting for observations: The symbolic level, the conceptual level and the subconceptual level [18, p. 204]:

Subconceptual sees observations are the firing of the neurons of our sensory receptors, without any conceptualization (*connectionism, black-box algorithms, ANNs, ML*).

Conceptual, where observations are defined not as token of a symbol, but as vector in a metric space of some quality (*prototype theory, linear algebra*).

Symbolic represents observations by describing them in some specified language (*formal logic, syllogisms, symbolism, classical AI, logical positivism*).

These levels are not in conflict, but different models of the same phenonemon, each covering distinct important aspects and each allowing a set of reasoning methods. The process of inducing a general rule from, few samples, for example is represented as pattern-matching on the firing patterns in the subconceptual level, which translates to the conceptual level as geometric reasoning through regions and direction. As another example, semantic relations such as hyponyms from the symbolic level are modelled as geometric sub-regions on the conceptual level.[9] Automatically generated conceptual spaces could allow to mimic high-level syllogistic reasoning from real-world data without the need to manually add countless facts. Additionally, it provides a new way to model reasoning and inference for both other levels through geometric relations, providing explanations for the noisy subconceptual level and computationally less complex algorithms for the symbolic level.

Summarised, regardless of the theory's aspiration to accurately model human conceptualization and reasoning, it provides a useful knowledge representation method and tool that allows to model kinds of human reasoning with novel algorithms that cannot be done with both other well-researched methods [18, Sec. 6.7]. Furthermore, it can serve as a representational format to express semantic relations for the semantic web [23] with a richer structure than classical ontologies (e. g. RDS, OWL or WordNet) and thus allows more than deductive reasoning based on strict *is-a* relationships and explicit, unambiguous, universal truths.

Definition

Conceptual Space. A conceptual space is a geometric structure used to encode the meaning of natural language terms, properties and concepts. The metric space is spanned by *quality dimensions* denoting basic domain-specific properties based on perception or subsymbolic processing. Natural language categories (*concepts*) correspond to convex regions, whereas points denote individual objects (instances/*entities*, allowing for geometric solutions to commonsense reasoning tasks such as *betweeness* or *induction*.

[9] For example, the validity of the statement *"a robin is a bird"* is given because the concept *robin* is geometrically a subregion of the concept *bird*.

In conceptual spaces, concepts are represented as convex regions in domain-specific, human-interpretable spaces. Figure 2.2 represents a sample space for the concept of *apple*, such that every instance of an apple is thus a vector that lies inside the region of the concept. This allows for high-level reasoning: The question *"Will an apple fit into my bag?"* can be answered by checking if the *size* dimension of the region is smaller than the dimension of the bag.

Formally defined, a conceptual space requires the following definitions:

Quality Dimensions are atomic units of perception. Some of these are necessarily linked (such as hue and saturation), making them *integral*, whereas others (e. g. temperature and weight) are *seperable*. Typically, each dimension corresponds to a primitive cognitive feature.

Domain A set of integral dimensions that are seperable from others, like the *color* domain made up from hue, saturation and value. Conceptual spaces are grouped into several low-dimensional subspaces according to these domains.

Similarity is defined as inverse distance, which requries a metric. A distinction can be made for the aggregation of integral and separable dimensions.

Betweenness An object Y is between two other objects X and Z if and only if d(x,y) + d(y,z) = d(x,z).

Natural Properties (*criterion P* [18]) are defined as convex regions of a domain in a conceptual space. A convex region has the property that an interpolation between any two points in this region is necessarily also in this region.

Concepts (*criterion C* [18]) are combinations of (potentially correlated) properties. "A concept is represented as a set of convex regions in a number of domains together with a prominence assignment to the domains and information about how the regions in different domains are correlated" [23, p. 8]

Entities are specific instances (tokens) of a concept, encoded as points.

Context can be modelled in a CS by weighting certain dimensions higher than others, influencing distance and how concepts are formed from properties.

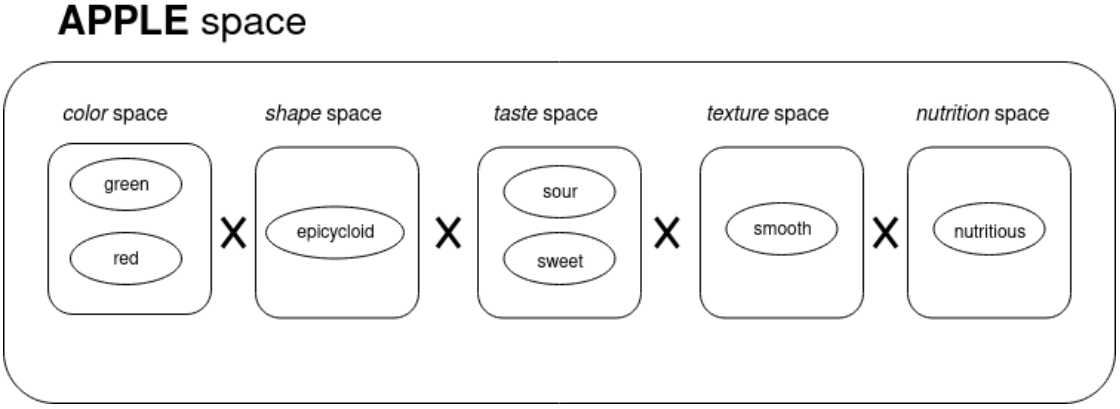

Figure 2.2.: Inner form of a Conceptual Space for an apple, displayed as product of different properties, which are convex regions in different quality domain spaces. Adapted from [24].

Let us explore some corollaries of the previous definition:

- Each conceptual space contains only items for which the space's dimensions make sense, so you would not find kings in a conceptual space of cabbages.
- *Concepts* roughly correspond to (non-proper) nouns. *Properties* generally encode Adjectives, and *points* refer to proper nouns (the name of e. g. a particular person, place, organization, or thing).
- From the criterion of convexity for natural properties and the definition of betweenness, it follows that if an object Y is between X and Z, and both X and Z have a property, Y must also have this property.
- Relative properties can be defined as regions on a relative scale - the property "*tall*" can accordingly be defined to be true if the entity is in the top 33% with respect to the size-property of all relevant objects.

2.3.1. Data-Driven Generation of Conceptual Spaces

So far, the area of application for Conceptual Spaces has been small. Most of the current works that rely on conceptual spaces create custom *phenonemal* spaces for semantic domains, where quality dimensions are chosen by the researchers (eg [25]).

The previous section has shown that CSs can serve as a framework for interpretable classifier that allow for explainable recommendation based on geometric reasoning, replacing the need to manually create knowledge bases. If, however, one needs to manually create these spaces, not much was gained. The work of [1] is a technique that allows to automatically generate spaces from pairwise dissimilarites of a corpus of texts in a data-driven fashion. In his book, Gärdenfors provided several suggestions how one could build spaces from high-dimensional input neurons. One of these was to use the MDS algorithm which creates a Euclidean space from pairwise distance matrices, such as an individual's assessement of similarities. The work of [1] basically follows this suggestion using classical AI algorithms.

To create the space, the authors unsupervisedly extract words of the text descriptions of corpus of entities of a specified domain. These words serve as candidates for semantic feature directions of a conceptual space. To find out which candidates are useful as features, they first embed all entities in a vector space. To identify which of the candidates constitute meaningful features, they create a linear classifier for each of the candidates that splits embeddings of descriptions that contain the word from those that do not. Those words for which the classification performance performs well enough are considered meaningful features.

In their paper, the authors create domain-specific conceptual spaces for three domains that allowed easy collection of text corpora, which were movies and their IMDB-reviews, placetypes and tags of photos at these places, as well as wines and their reviews on a respective platform. Each movie, place-type or wine will henceforth be termed entity. A representation of a movie is then generated from the BoW of the descriptions of the individual movies, leading to a very high-dimensional and sparse representation for all movies. To make the representations less sparse and more meaningful, the words in the BoW are subsequently PPMI-weighted, which (similar to tf-idf) weights words that

appear often in the description of a particular movie while being infrequent in the corpus overall high, ensuring that discriminative words are more relevant in the embedding.This weighted BoW is, however, no Euclidean space, which is why the authors subsequently use MDS, a dimensionality reduction technique that creates a Euclidean space while ensuring that original distances are preserved as well as possible. This space already allows for interpretable geometric reasoning such as betweeness, but its directions are not interpretable human concepts. To find these, the authors assume that words that describe relevant features of the respective entites appear among their descriptions and that words describing meaningful features correlate with good classifier performance in separating entities based on them. To classify, the authors then use linear classifiers such as SVMs.

Consider the domain of movies, and the word "scary" as candidate feature direction. The movie-embeddings are grouped into those that contain the words and those that do no, and finds a hyperplane that divides both groups. The advantage of linear classifiers is that they create a linear hyperplane that best separates positive from negative entities. The orthogonal of that hyperplane is a vector, which can serve as feature axis: The distance of orthogonally projecting an entity onto this vector induces a ranking of entities. The further away an entity's embedding is from the decision surface on the positive side, the more this feature applies to the entity. To assess the performance, [1] use Cohen's kappa measure to compare the ranking induced by the plane with the number of occurences of the word for the entity. The better these rankings match, the higher the likelyhood that a good feature direction was found. To reduce the number of resulting features, they are subsequently clustered based on the similarity of their orthogonals before the mean direction of this feature-cluster is calculated.

The final **feature-based representation** is generated by representing each entity as vector whose individual components corespond to the ranking of this entity compared to all others for each of the most salient feature directions. Thus, in the final feature-based representation only the relation of each entity in relation to all others with respect to the salient directions is relevant. Given that respective directions are not orthogonal [1, p. 22] and that rankings are only ordinal (where distances are not quantifiable), the final embedding loses some geometric properties such as the Euclidean distance metric, but gains interpretable directions.

The algorithm is optimized to *look good to humans*, meaning there are no straight-forward metrics or obvious evaluations. To evaluate its performance quantitatively, we will test if it is possible to use the detected semantic directions to classify human concepts among the data, such as the genre assigned to a movie.

2.3.2. Explainable Reasoning with Conceptual Spaces

The goal of this thesis is to provide explainable recommendation for educational resources. The main reason we are considering CSs as knowledge representation method for our domain is their ability to model many cognitive reasoning methods computationally. So now, after having explored what conecptual spaces are and how to generate them, let us see how they help to allow for explanable reasoning. The relations of semantic features

can be used for classification based on properties understanable for humans. Further, they allow the entities to be ranked such that a user can specify her demanded degree how much a property applies, as exemplified by the Movie Tuner in Figure 1.1.

Conceptual Spaces allow to model many forms of commonsense reasoning geometrically or algebraically. Another property that is important to us is that in contrast to typical recommendation engines, in CSs similarity is necessarily context-dependent. Derrac & Schockaert [1] use these features to create many forms of reasoning systems such as ones based on betweeness, *relational similarity* or *a fortiori rasoning*. Let us look at what this means in practice, as well as how standard symbolisic reasoning can be modelled in conceptual spaces.

Categories and Ontologies

Logic-Based reasoning/inference can model many forms of reasonings, but it requires the knowledge to be encoded in logic in manual work and doesn't allow for fuzzyness. In formal logic/ontologies/lexical databases, semantic relations of concepts are explicitly modelled. The RCC [26] links these relations to their geometric interpretation, providing a bridge between these and Conceptual Spaces [27], in which natural language terms are not modelled as points or vectors, but as convex regions Once the handled entities are embedded into a conceptual space, many ontological relations are implied through the relation of these regions. This allows to replace symbolic inference engines with geometric algorithm when using the richer structure of CS instead of ontologies. Table 2.1 displays these respective relations. Apart from these however, a CS encodes more than just a taxonomy of concepts, but allows for interpolation and extrapolation of knowledge. This allows for other human forms of resoning.

Ontology Relation	Other Names	RCC5 [26] analog / *Geometric equivalent*	Example
Type Identity	Equality of Concepts, Synonymy	Identical Regions (EQ)	Animals with a liver & Animals with a heart
Subsumption	Hyponyms/ Hypernyms, *is-a-relationship*, Concept Hierachies, Taxonomies	Proper Parts(PP, PP^{-1}), *Subregions*	*Every pizzaria is a restaurant*
Mutual Exclusiveness		Discrete Regions (DR), *unconnected/disjoint regions*	*No Restaurant can be a beach*
Overlapping Concepts		Partial Overlap (PO)	*Some bars serve wine, but not all*
Opposites		*Set inverse*	Humans & Non-human animals
Token Identity	Equality of Names, Synonymy	*Equal coordinates*	Morning star & Venus
Meronymy / Holonymy	*part-whole realationship*	*Product spaces* (see [24]	Trees & Leaves

Table 2.1.: Link between Conceptual Spaces, the Region Connection Calulus and Ontologial Relations

Similarity-based reasoning

As discussed in Section 1.2.1, modern recommendation algorithms often rely on similarity-based reasoning by suggesting that users that liked and item may also like similar items (collaborative filtering). In terms of classification, this corresponds to the 1-Neirest-Neighbor approach, where an object is assigned the class of the most similar item:

Alice likes *the Lord of the Rings*
The Hobbit and *Lord of the Rings* are similar

Alice will probably like *The Hobbit* $\therefore$

A has property **x**
B is similar to **A**

B likely also has property **x** $\therefore$

This cannot be done with classical logic, which can not model degrees of something but only universal full truths. While modern recommendation engines require only labelled examples, these algorithm lack *explainability*: The employed distance function does not encode *in what respect* two items are similar. For human reasoning however, there is no such *overall Similarity* - instead similarity is relative to a domain and only meaningful in context[18, p. 110]. In a conceptual space, this can be modelled: The distance function can give weight for certain dimensions depending on context or objects of different concepts can be considered similar if they share enough properties. Most importantly however, in a CS a system for recommendation can ask users what dimensions of a given entity the user liked to suggest items that are similar in that regard.

Induction

Another very important tool of human common-sense reasoning is abstraction and generalization: Going from single observations to general rules. For that, we need to decide which properties of the respective observation are relevant, distilling sensible information from irrelevant noise. Instead of just implementing *black boxes* that do inexplicable pattern matching, it helps to look at the underlying rules. On the Conceptual Level, eg *If two different entities of the same class have a property, maybe all entities of that class have a property* can be modeled through *categorical inductive inverences*:

Grizzly bears love onions
Polar bears love onions

All bears love onions [18, p. 226] $\therefore$

According to Gärdenfors [18], representations do not need to be similiar to the objects they represent, but the *similarity relations of the representations* should correspond to those of the objects they represent. From this follow a few other translations of reasoning and geometric properties. [1] elaborate on many other of these and build semantic classifier that model other kinds of reasoning. For this thesis, let us look at one example of these, which may prove useful in recommendation and classificiation and exemplifies how the representations benefit from salient directions, namely **A fortiori reasoning**:

The shining is a horror film
The shining is scary
It is more scary than *The shining*

It is likely also a horror film $\therefore$

A has property **x**
B is *more severe* than **A**

B likely also has property **x**
or a *more severe* property $\therefore$

As argued by Derrac & Schockaert [1], automating this and many other forms of inference requires a richer form of knowledge than what is available in classical logic, namely a notion of **Betweenness** and **Directionality**. In Euclidean spaces, the intuitive notion of these concepts is most faithfully modelled. Because of that, the authors regard it a necessary condition for their algorithm.

Summarised, conceptual spaces allow to model the forms of explainable reasoning that thus far required symbolic inference engines and do not share the problems of symbolistic approaches such as having to explicitly identify synonymy. On top of that, they also to allow to model fuzzy real-world concepts and relations. If Euclidean spaces are used, similarity relations of the representations, such as betweeness and parallelism, correspond to the real-word objects they model.

2.4. Other Related Work

This thesis focuses on the aforementioned algorithm, primarily considering [1] and on top of that only two follow-up works: [2, 3], which have shown to provide useful extensions for it without changing its core logic. This shall by no means mean that these are the only ones that could be considered.

Tag Genome By far the closest to what we do is the algorithm of Vig *et al.* [16], who generate a so-called tag genome for the domain of movies supervisedly based on keywords that users have assigned manually. Their algorithm takes these binary assignments and creates a dense representation that encodes a degree of relevance for each combination of movie and tag. Furthermore, they create a dedicated movie recommendation system on basis of this (interface reprinted in Figure 1.1). This system provides explainable recommendation based on these tags, allowing users to request recommendations for movies such as *"I'd like something less violent than Reservoir Dogs"* [16, p. 3]. Not only is their application exactly what is being demanded here, but the algorithm itself also performed significantly better than the one of Derrac & Schockaert [1] in a human study of the latter, in which they directly compared the techniques by asking subjects which of the respectively extracted keywords better describes the difference between two movies [1, p. 44]. Considering however that Vig *et al.*'s [16] algorithm is supervised and requires data which does not exist for our domain, it cannot be applied in our case. On the contrary, the final results of their algorithm are preferred by users over the ones generated with the algorithm considered in this work, but structurally exactly equal. This provides clear evidence that the desired application of this work is possible, albeit of lower quality than what their work achieved.

Generally, what is done here corresponds to **Representation Learning**, whose aim is to discover the inherent semantic structure of a representation unsupervisedly [28]. More specifically **Disentangled Representation Learning**, where only salient attributes relevant to the task at hand should be extracted, which means finding latent embeddings whose dimensions are meaningful interpretable features. Generative Adversarial Networks [29] or Variational Autoencoders [30] are modern techniques that are good at finding latent information in images. Especially InfoGAN [31] should be named, which

can extract interpretable features such as pose, hairstyle, prensence of glasses and emotions from images unsupervisedly.

LDA In the realm of NLP, this also relates to **Topic Modeling**, which aims to extract multiple hidden themes from a given text corpus by discovering groups of co-occuring words unsupervisedly. A well-known algorithm for this is Latent Dirichlet Allocation (LDA) [32], which represents documents by its salient *topics*, each of which being a cluster of natural language terms. This technique bases on the assumption that each text consists of various topics, which are in turn made up by various keywords, making it possible to represent texts as multinomial distribution over latent topics which are aggregations of these keywords. Assuming a hierachical bayesian distribution where each text of a corpus is represented as mixture of topics it contains, their unsupervised algorithm extracts these by approximating the underlying infinite mixture of topic with an expectation-maximization (EM) algorithm. This yields a representation where each text is explicitly represented by the most propable words according to this distribution for a finite number of most probable topics. The algorithm finds use in text classification and collaborative filtering, but relies on unflexible BoW representations, making it hard to incorporate additional information such as correlations between topics [2].

Academic Interests Recommender Regarding the used domain, there is already a system incorporated into the Siddata-DSA that aids students by finding and recommending educational resources. SidBERT [33] extracts implicit information from courses and other learning material by their title by categorizing them into one of 905 classes derived from the third or fourth level of the DDC [34], a hierachical tree stucture system commonly used to categorize library books. SidBERT uses the same dataset as this work and classifies with a custom classification head ontop of a BERT-encoder ANN which is trained on 1.3 million book titles collected from three universities as well as the German National Library, currently achieving 45.2% test accuracy (62.2% recall) among 905 classes.

Variations of this Algorithm There are also techniques that extend the algorithm of Derrac & Schockaert [1]: *Alshaikh et al.* [35, 36] use this algorithm as one of their steps and create a similar algorithm to find disentangled features that is in line with the requirement of Conceptual Spaces to consist of low-dimensional domain-specific subspaces. Regarding other unsupervised ways to create Conceptual Spaces, Gärdenfors himself suggested in his book [18] to use self-organizing maps (Kohonen-Networks [37]) instead of classical NLP algorithms and MDS to unsupervisedly create concpetual spaces. Finally, the whole concepts of vector-space models for words [38] and texts [14, 15] is related in that represents the meaing of terms, phrases or documents by embedding them in a vector space. However these have arbitrary non-interpretable dimensions and are no metric spaces, thus having no relation of geometry and meaning for e. g. betweeness or analogical reasoning, which will be eloborated in the next section. For more related work it is also referred to the respective sections of [1–3].

2.5. Relevant Algorithms and Techniques

Thus far, we have described the base algorithm which this thesis replicates. Before describing each of its step in detail, it is useful to get a grasp of the theoretical foundation of the creation of linguistical VSMs in general. This also helps to place the algorithm in the context of the field of computational NLP. Note that this section primarily serves to understand the key concepts required for the methodology of Derrac & Schockaert [1]. To do that, we will sometimes put emphasis on other algorithms of the same type than those actually used in the implementation if they express the model more explicitly. The methodology is modular and e. g. only requires *some* algorithm for dimensionality reduction.

2.5.1. Classical Vector Space Construction

The methodology in question consists of several components, each of it being an algorithm in itself. The first steps are basically classical linguistic tools: A text corpus is preprocessed and the words of its entites are counted and raw counts are transformed. From these values a frequency matrix is generated and its dimensionality is reduced, before directions for domain-specific similarity measures between the regarded entities are extracted.

Lowe [39] conceived a general framework to construct vector spaces from texts, splitting the process into the steps of first counting the token frequencies, then transforming the raw counts into more useful quantificiation measures and smoothing the space using dimensionality reduction, before calculating the similarities on the resulting embedding. While the considered algorithm requires more steps before calculating the similarities to allow for domain-specific and other more complex forms of reasoning, it also adheres to this structure.

Distributional Semantics

> *"you shall know a word by the company it keeps"* - Firth [40]

The core principle behind any kind of embedding of linguistics units such as words or documents is the distributional hypothesis, which states that linguistic items with similar distributions have similar meanings. If this is the case, the meaning of words correlate with their distribution: Words that occur in similar surroundings have similar meanings.

More precisely, vector-space models fall into different categories depending on if the similarity of documents (*Term-Document-Model*) or of words (*Word-Context-Model*) is in question. Because the algorithm considered here describes the relation of entites through their descriptions it falls under the latter category. According to Turney & Pantel [41], the assumption underneath the Term-Document model is more precisely called the **bag-of-word hypothesis**, which states that documents with similar distributions of words have similar meaning. Accordingly, when embedding documents into a vector space, it must be ensured that the similarity relations of the embeddings closely resemble the similarity of the original documents.

Many NLP tasks rely on documents being represented as vectors, such as Information Retrieval, Recommendation, Text Classification, Translation, Sentiment Analysis and

many more [11, 13–15, 41–45]. The process of turning a collection of texts document into numerical feature vectors is referred to as *vectorization*. According to Turney & Pantel [41] (who base their work on Lowe [39]), the construction of a VSM from texts can be decomposed into a four steps:[10]

1) Building the Frequency Matrix which starts with preprocessing such as tokenisation followed by normalizing and possibly lemmatizing the tokens amongst many other possible techniques, before counting frequencies of either words or n-grams, yielding a matrix of BoWs.

2) Transforming Raw Frequency Counts "[B]ecause common words will have high frequencies, yet they are less informative than rare words" [41], it may make sense to adjust the weights of the elements of the frequency matrix such that the distances are not distorted by them.

3) Smoothing the Frequency Matrix A matrix that counts the frequency of any word in the corpus is generally noisy, sparse and extremely high-dimensional. Dimensionality Reduction helps to counter all three issues, yielding vectors closer resembling the document's *latent information*.

4) Calculating Similarities of individual vectors is the final step and aim of most embedding algorithms. This can be done in various ways, a classical technique is to use their cosine distance.

Importantly, our algorithm differs from this four-step-process by injecting several additional steps before calculating similarities. This is because we hold the notion that similarity is necessarily context-dependent and there is no overall similarity (see Section 2.3.2), which requires additional dissection of the final step. In the following, we will describe three steps relevant for us.

1. Bag-of-ngrams representation

The most relevant information that can be taken into account when comparing two texts are the words they consist of. Accordingly, an obvious choice to vectorize a collection of documents is to describe each document by the counts of its word occurences, which is called BoW-representation.

This approach is simple but has important drawbacks: Firstly, a document is not fully described only by the words it contains but also by their constellation. The information about ordering and relative position is lost in this representation (consider not knowing the position of negations). To alleviate this issue, texts can instead be represented as glspln-grams, where the tokens are not single words but all sequences of words of length n. The drawback is that this vastly increases sparseness and dimensionality.

Further, by representing each word as separate *one-hot-vector* ignores word semantics: as every vector is equally distant from any other, synonyms (cases where the same meaning is expressed with different words) are just as far apart as antonyms (opposites). This unreliability of term-document-association is what Deerwester *et al.* [46] calls the *fundamental information retrival problem*: The mapping between words and their meanings is

[10] When considering neural embeddings such as Word2Vec, the separation of these steps is hidden in the algorithm and not as distinct, but the principles hold also in these techniques.

not bijective, but ambiguous. Different words can be used to express the same concept, and sometimes the same word refers to different concepts. This issue is partially alleviated by the dimensionality reduction performed subsequently, which may provide a way to determine what concepts are implied and obfuscated by fallible word choice.

To address both issues, modern algorithms exist, which train a document embedding directly based on constellation (alleviating lost ordering) of word-embeddings that are pre-trained based on their usual context (alleviating distance relation problems)

2. Word-weighting techniques

When comparing the BoW-representations of texts, it is reasonable to give more weight to *surprising* words than to expected ones. The idea behind that is, that "surprising events, if shared by two vectors, are more discriminative of the similarity between the vectors than less surprising events." [41, p. 156] Another crucial reason is, that individual texts in the corpus are of drastically varying length, so longer entities would naturally dominate shorter ones when only comparing the raw counts - considering relative frequencies instead of absolute ones alleviates such problems. The algorithms explained below transform the raw frequency-counts of a document and an n-gram into some *score*, dependent on the number of occurences of this term in this document as well as the counts of other n-grams and other documents. This score is henceforth called a quantificiation.

Let us consider term t, corpus C, document $d \in C$ (represented as BoW). Then:
term-frequency $\mathrm{tf}_{t,d}$: How often t occurs in d
document-frequency df_t: How many documents $\in C$ contain t
summed term-frequency $\mathrm{df}_{t,*} = \sum_{d' \in C} \mathrm{tf}_{t,d'}$: How often t occurs in any document $\in C$

Tf-Idf The most well-known technique formalizing this concept is tf-idf, which gives a term-document pair a higher weight if the term is generally rare in the corpus (low df) and frequent in the respective document (high tf):

$$w_{t,d} = tf_{t,d} * log(\frac{|C|}{df_t})$$

PPMI Turney & Pantel [41] suggested to use the PPMI measure instead of tf-idf to weight the counts in document-term matrices, relying on [47]'s work taking into account psychological models to extract information about lexical semantics from co-occurence statistics. According to these works, PPMI performs most plausible when measuring semantic similarity in word-context matrices compared to human evaluation. For that reason, Derrac & Schockaert [1] and its follow-up works [2, 3] rely solely on this technique. Like tf-idf, it weights terms that are strongly associated with a document highly by favoring terms frequently associated with document d while infrequent in the corpus overall. For that, it uses the logarithm of the probability of the considered term-document-combination d, t, normalized by the probability of this document co-occuring with any term $(d, *)$ and this term co-occuring with any document $(t, *)$

$$w_{t,d} = max\left(0, log\left(\frac{p_{d,t}}{\sum_{t'} p_{d,t'} * \sum_{d'} p_{d',t}}\right)\right)$$

$$p_{d,t} = \frac{\mathrm{tf}_{t,d}}{\sum_{d'} \sum_{t'} \mathrm{tf}_{t',d'}}$$

3. Dimensionality Reduction and Latent Space Embedding

At this step, we have a sparse and high-diensional matrix quantifying the importants of all corpus-terms for all documents. The next step is to smooth this frequency matrix. Reducing the dimensionality of this matrix reduces computational processing load and helps in alleviating the prevalence of irrelevant noise in the original matrix [41]. Additionally, the right algorithms may also make use of the principles of distributional semantics to alleviate the aforementioned problems of word distance and synonymy. The result is that the document vectors depend less on their exact phrasing and instead more closely resemble the concepts that these words expressed, also called its *latent* (hidden) *topics*. This may even improve subsequent similarity measurements, as it is less distorted by noise and word choice irregularities.

The technique that most explicitly models a document's latent topics is LSI [46], which relies on the fact that words that are close in meaning will occur in similar pieces of text to yield embeddings where words and documents of similar meaning are similar.

LSI/LSA[11] While language is used to express the world, it is also highly ambiguous and redundant, such that the relationship of model and reality is only a statistical one. The same thing can be expressed with different words (synonymy), and sometimes a word has different meaning depending on context (polysemy). The underyling latent semantic structure in texts is obscured by the randomness of word choce, such that individual words provide only unreliable evidence about the conceptual topic or meaning of a document.

However, language has a lot of structure, which allows to treat this as statistical problem: The occurrence of some patterns of words gives a strong clue as to the likely occurrence of others. According to the distributional hypothesis, the matrix of observed occurrences of terms applied to documents can be used to estimate parameters of the underlying true model. This is explicitly done using linear algebra operations on the frequency matrix in LSI [46].

This algorithm decomposes the document-term matrix into a product of three linearly independent factors: *words-per-topic* $\times$*topic-importance* $\times$*corpus-topic-distribution*. Then it runs *Truncated Singular Value Decomposition* (SVD), which finds a lower-rank approximation of the matrix containing the *hidden information* while identifying and keeping relationship patterns. In other words, by taking only the important components from this matrix (*rank-reduction*), the new representation becomes lower-dimensional, less noisy and less sparse, but still approximates the original similarity behaviours.

LSI explicitly represents both documents and terms in the same semantic space, such that they are treated equally when the SVD analyses their similarity behaviour. It yields a representation of arbitrary dimensionality that captures the relation of term-term, term-document and document-document similarity by ensuring that similar items will end up close in the space. Importantly, this yields a representation in wich both documents and terms correspond to vectors. The similarity of each of these is assessed via the

[11] Both terms refer to the same algorithm, which is generally called LSI when applied to document similarity and information retrieval, and LSA when applied to word similarity.

cosine-distance. To analyse the similarity of a document and a query term, the term is embedded by generating *pseudo-document* (a document that contains only this term), and close vectors are returned, which may include results that conceptually similar but do not share any words.

As consequence of the compression, some dimensions are combined and depend on more than one term - generally this merges embeddings of similar terms, mitigating synonymy. Also Polysemy is reduced, as only the components of polysemous words that have encode a similar meaning than the cluster are relevant for the combined direction of the vector. Another consequence is that terms that did not actually appear in a document may still end up close to the document, if that is consistent with the major patterns of association in the data: Terms that *could* have been used as well.“if the tags *love story* and *hilarious* have been applied to an item, it is likely that the tag *romantic comedy* is highly relevant[16].”

The algorithm to create the Tag Genome in [16] bases to a high degree on this algorithm: It creates a binary frequency matrix for all considered documents and a set of *tags*, embeds documents and pseudo-documents generated from the tags and compresses it with LSI. The relevance of a tag for a documents `tag-lsi-sim(t,i)` is then calculated by their embeddings' cosine similiarity. Also, the algorithm appears to be a good method to find clusters of similar terms for semantic direction, given that it considers these terms as directional vectors already.

Multi-Dimensional Scaling Derrac & Schockaert [1], require that spatial relations such as betweeness and parallelism hold in the vector space embedding. For that, they require a space of Euclidean metric which is not given in LSA, which generates a space based on the similarity-centred objective to reduce cosine distances of similar entites. Instead they follow Gärdenfors [18] suggestion and rely on MDS. While the algorithm bases on Euclidean distance, the principle that items of similar meaning will end up in similar positions still holds. In fact, it should hold most kind of compression, bearing in mind the distributional hypothesis and the fact that if there is some inherent *structure* (obfuscated by wording) that explains the dissimilarity of any text corpus sufficiently well, it will remain after the compression.

MDS [48] is a dimensionality reduction technique that induces a finite vector-space representation from pairwise similarities. It takes a dissimilarity matrix as input and returns a fixed-dimensional embedding in which original distances are kept as close to the ones of the dissimilarity matrix as possible. Gärdenfors considered using this technique it to find an underlying *phenonemal* conceptual space from human similarity judgements, given that it yield qas high a correlation as possible between the similarity judgements of the subjects and the corresponding distances in the estimated dimensional space [18, p. 22]. The algorithm has quadratic space complexity [36]

Especially relevant is the *metric MDS* algorithm, which requires metric distances as input and models their similarity as distances in a space finite space of lower dimensionality with a Euclidean metric. A distance measure is a metric if the following axioms hold:

$$d(x, y) = 0 \Leftrightarrow x = y \tag{2.1}$$

$$d(x, y) = d(y, x) \tag{2.2}$$

$$d(x, t) \leq d(x, y) + d(y, z) \text{ (triangle inequality)} \tag{2.3}$$

Given metric distances $d(p_i, p_j) \forall 1 \leq i, i \leq j$ and the demanded number of dimensions k, the algorithm starts with a random initial distribution and progressively adjusts the coordinates by re-computing the embeddings $v_1, ..., v_n \in \mathbb{R}^k$ such that distances are maintained as well as possible by minimizing

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (d(p_i, p_j) - \|v_i - v_j\|)^2$$

3. Methods

This chapter explains the methods that were used to achieve the two goals of replicating the algorithm of Derrac & Schockaert [1] with the improvements of [2, 3] with a sustainable architecture and applying it to the domain of educational resources.

To do that, we will firstly look at the datasets used by [1–3], as well as our own one and compare key statistics. The subsequent section then explains the used algorithm broadly before looking at each of its steps in detail. Based on that, the specific architecture that was created as well as the workflow used to generate meaningful results will be introduced to demonstrate its process. The final section explains what kind of methods and metrics were used to generate and interpret the results that will follow in the next chapter.

Given that there were two research questions, one asking if the replicating algorithm can be applied to another domain and the other asking for a reliable application, it makes sense to both look at datasets and general algorithm on the one side, but also at the worflow and architecture. Writing about the latter while also open-sourcing the codebase is especially useful to ensure ease and speed of future replication, such that all claims can be independently tested with the exact same implementation without having to rely on ambiguous and incomplete verbal explanations.[1]

3.1. Datasets

Let us first elaborate on the datasets used both in the works that are being replicated as well as the dataset used in the scope of this thesis. Doing both is important because (as explained in Section 2.1.1), to really know if an algorithm is a good choice for a task, it must be ensured that it does not only work on a special kind of dataset due to a special property that it happens to have. So, we will first compare all datasets to see if there are important differences between them, as for example in its structure, size or just general *logic*.

Derrac & Schockaert [1] used their algorithm to create conceptual spaces for three domains: *movies*, *placetypes* and *wines*. Accordingly, they created corpora for these three domains. The two considered follow-up works both re-used the datasets for movies and placetypes and also created additional datasets respectively. Table 3.3 shows a complete comparison of all datasets used in any of [1–3], including their origin, properties as well as associated pre-processing steps.

Comparing dataset properties

This work applies the replicated algorithm to a new dataset, so it is important to check if the new dataset differs from the originally used ones. To ensure comparability in algorithm details, this implementation is also run on the *placetypes*-dataset, and some of its statistics as well as that of the other two datasets made public by [1], available at

[1] To achieve reproducibility. Just like this thesis could have been a lot less effort if [1–3] did that instead of ambiguous incomplete descriptions.

https://www.cs.cf.ac.uk/semanticspaces/. Let us first compare some key statistics of them, before looking into two of them individually in the next section.

A very distinctive difference of the datasets is their size: the movies-dataset consists of 15.000 entities, placetypes of 1383, and wines of only 330. Just as important as the number of entities is however also the length of their associated text-corpora. Figures 3.2 and 3.4 show a histogram of the distribution of text lengths for the Siddata-dataset and placetypes-dataset respectively, and Table 3.1 summarise corpus size and distribution of text lengths per entity for all handled datasets. Note that the Siddata-dataset is listed twice, once including all available entities and once after filtering out those whose description was shorter than 80 words.

	Entities	**Words per Entity**					
		Unique			**Non-Unique**		
		5th	**Med**	**95th**	**5th**	**Med**	**95th**
movie reviews	13 978	565	1358	5 510	962	3179	38 378
place types	1383	159	2215	18 117	2378	55 422	886 233
Siddata (all)	26 346	19	60	169	22	71	239
Siddata ($\geq$ 80 words)	11 601	63	99	211	83	124	323

Table 3.1.: Sizes of the text-corpora of the Siddata-dataset and those of [1]. The columns *Words per Entity* list thresholds for the 5^{th}, 50^{th}, and 95^{th} percentile.

	1	**2**	**5**	**25**	**50**	**100**	**500**	**1000**	$\lvert C\rvert$**/10**	$\lvert C\rvert$**/4**
Siddata	163 285	80 802	35 697	9702	5535	3059	570	210	23	2
placetypes	746 180	428 810	182 906	41 320	21 833	11 166	1452	183	8047	2669
movies	589 727	279 429	128 850	55 429	39 976	28 768	11 431	6931	1786	332

Table 3.2.: Words that exceed various df thresholds for the Siddata, placetypes and movies datasets. In [1], the candidate-threshold for placetypes was 50, and the threshold for movies 100. The last columns are relative to the dataset-size ($|C|$ = number of entities).

It is obvious that even though the number of Siddata-entities is comparable to the bigger of [1]'s datasets, the number of unique words in the individual texts associated with the entities are two orders of magnitude shorter than both. As the algorithm does not rely on deep learning, this does not mean that the dataset is unsuited for it, but let us explore how the distance in their lengths can be explained and what consequences this has.

The works considered in this replication [1–3] all use the movies- and the placetypes-dataset to evaluate their methods. The former consists of the concatenation of all available reviews for 15.000 movies from IMDB,[2] grouped by the movie the reviews are for. The placetypes-dataset is created from a collection of tags that belong to photos uploaded to the photo sharing platform Flickr[3] that co-occur with other tags that denote one of several placetypes.

[2] https://www.imdb.com/
[3] https://www.flickr.com

dataset	contents	preprocessing	size	classification classes	candidate word threshold
movies[4] **[1–3]**	grouped-by-movie-concatenated reviews for movies	• removed stop-words[5] • lower-cased text • removed diacritics • removed punctuation	[1]: 15000 movies [2, 3]: 13978 movies	• genre (23 classes) • plot keywords (eg. *suicide, beach*) (100 cls) • age-rating certificates (6 classes)	df $\geq$ 100 $\rightarrow$22 903 candidates variable-length **n-grams** considered
place types[4] **[1–3]**	Tags of Flickr-photos that are also tagged with a place-type	None	1383 place types	• category from GeoNames (7 classes) • category from Foursquare (9 classes) • category from OpenCYC (93[1]/20[2, 3] cls)	df $\geq$ 50 $\rightarrow$21 833 candidates (all words from the BoW) **n-grams**: squashed all words of a tag
wines[46] **[1]**	grouped-by-wine-variant-concatenated reviews for wines	• removed stop-words[5] • lower-cased text • removed diacritics • removed punctuation	330 wine-varieties	*not performed*	df $\geq$ 50 $\rightarrow$around 6k candidates variable-length **n-grams** considered
20 newsgroups[7] **[2]**	posts partitioned roughly even across 20 different newsgroups	• Headers, footers and quote metadata removed[8] • removed stopwords (using NLTK's corpus [49]) • lowercased text • candidate terms: all textual and numerical tokens	18446 posts	• newsgroup post belongs to (20 cls)	$\geq$ 30 occurences
imdb sentiment[9] **[2]**	highly polar movie reviews for binary sentiment classification	• removed stopwords (using NLTK's corpus [49]) • lowercased text • candidate terms: all textual and numerical tokens	50000 reviews	• sentiment of the review (2 classes)	$\geq$ 50 occurences
Bands [3]	All Wikipedia pages ($\geq$ 200 words) whose WikiData semantic type is "Band"	• removed HTML-tags and references • *"standard preprocessing strategy"* [35, p. 137] • removed stopwords (using NLTK's corpus [49]) • POS-tagging and keeping only nouns and adjectives • remove words with a rel. df $>$ 60% or abs. df $<$ 10	11448 bands	• Genres (22 classes) • Country of origin (6 classes) • Loc. of formation (4 classes)	10 $<$ df $<$ 6869 (all words from the BoW)
Organisations[10] **[3]**	All Wikipedia pages ($\geq$ 200 words) whose WikiData semantic type is "Organisation"	• removed HTML-tags and references • *"standard preprocessing strategy"* [35, p. 137] • removed stopwords (using NLTK's corpus [49]) • POS-tagging and keeping only nouns and adjectives • remove words with a rel. df $>$ 60% or abs. df $<$ 10	11800 organisations	• Country (4 classes) • Headquarter Loc. (2 classes)	10 $<$ df $<$ 7080 (all words from the BoW)
Buildings[10] **[3]**	All Wikipedia pages ($\geq$ 200 words) whose WikiData semantic type is "Building"	• removed HTML-tags and references • *"standard preprocessing strategy"* [35, p. 137] • removed stopwords (using NLTK's corpus [49]) • POS-tagging and keeping only nouns and adjectives • remove words with a rel. df $>$ 60% or abs. df $<$ 10	3721 buildings	• Country (2 classes) • Administrative loc. (2 classes)	10 $<$ df $<$ 2233 (all words from the BoW)

Table 3.3.: All datasets used by any of [1–3]. Citations behind the dataset name denote which author used it. Other listed properties include dataset sources (where available), contents, sizes, the respectively used preprocessing-methods and candidate-word-thresholds, as well as the classes considered in the evaluation of the derived explainable classifiers.

[4] `https://www.cs.cf.ac.uk/semanticspaces/`
[5] `http://snowball.tartarus.org/algorithms/english/stop.txt`
[6] `https://snap.stanford.edu/data/web-CellarTracker.html`
[7] `http://qwone.com/~jason/20Newsgroups`
[8] Using the scikit-learn python package, see `https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html`
[9] `http://ai.stanford.edu/~amaas/data/sentiment/` [45]
[10] Originally created in and for [35]

Other datasets considered in the works of [1–3] include wine reviews, posts to certain newsgroups, and another dataset created from IMDB-reviews (see Table 3.3).

All of these datasets have in common that they are made up from a collection of independent texts or tags, created by different people. This means, that the more obvious or distinct a property of the respective entity is, the more often words describing that property will be used as tag or as part of the review. For example, a movie that is *scary* to a lot of people will lead to many reviews mentioning that, which means that the word scary (or other words commonly co-occuring with it) will have a high count in the concatenation of all these reviews. The algorithm from [1] heavily leans on this property by using the (relative) frequency of certain words as signal for the importance of the concept they may refer to.

The Siddata-dataset unfortunately does not share this property, as the texts that belong to an entity are not collected from different independent texts, but solely from the description of that entity. It may of course be roughly the case that the more *mathematical* a course is, the more often the word *math* occurs in its description, but the correlation is likely not as prominent as in the aforementioned datasets. It should however be noted that [3] also used the algorithm on three datasets where the description for an entity was collected from its Wikipedia[11]-article, which also does not contain the described duplication of words.

Other considered datasets One of the goals was to build an adaptable architecture, so we also want to check if it can quickly and easily be applied to other datasets. For that, a dataset of 1002 short stories from project guttenberg[12] and one of 100 000 coursera course reviews[13] were created. The former is similar to our dataset because it is also not made up as concatenation of multiple invididual texts. The latter is made up from reviews like the aformentioned ones, but from the educational domain. These were created primarily to test the architecture, and no results will reported or analysed for these datasets, as that would go far beyond its scope and intended length.

3.1.1. Siddata-courses

The main goal of this thesis was to create a conceptual space of courses, automatically generated by course descriptions. For that, a dataset of courses and their descriptions was obtained as export from the Stud.IP system as used at the universities of Osnabrück, Hannover and Bremen.

Resource Type and Origin

As explained in Section 2.2, the dataset itself is collected collected from all courses that are present in the Stud.IP systems of the universities of Osnabrück, Hannover and Bremen and is crawled by the Siddata-DSA. Additionally, it contains a small amount of MOOCs and other OERs and resources collected by web crawers. The distribution of types (if an entity is collected from the LMS, an OER or another web source) is depicted

[11] `https://en.wikipedia.org/`

[12] `https://www.kaggle.com/datasets/shubchat/1002-short-stories-from-project-guttenberg`

[13] Source: `https://www.kaggle.com/septa97/100k-courseras-course-reviews-dataset`, Exploratory Analysis: `https://www.kaggle.com/roshansharma/coursera-course-reviews`

in Figure 3.1 among the distribution of other metadata. Almost all educational resources in the dataset are *Courses*, which refers to regular events at the respective university. As the figure shows, there are also PDF-documents in the dataset, but again the amount is neglibile. Roughly 75% of all resources in this dump are courses from UOS, another 23% courses from the universities Bremen and Hannover, the rest split among other formats.

The data used here is collected from three separate dumps, created at different protype stages of the DSA. The dumps had a high overlap in resources, however had different metadata attached to them, requiring careful merging for the creation of the individual raw datasets. So far, neither the datasets nor any of its underlying dumps have been published.[14]

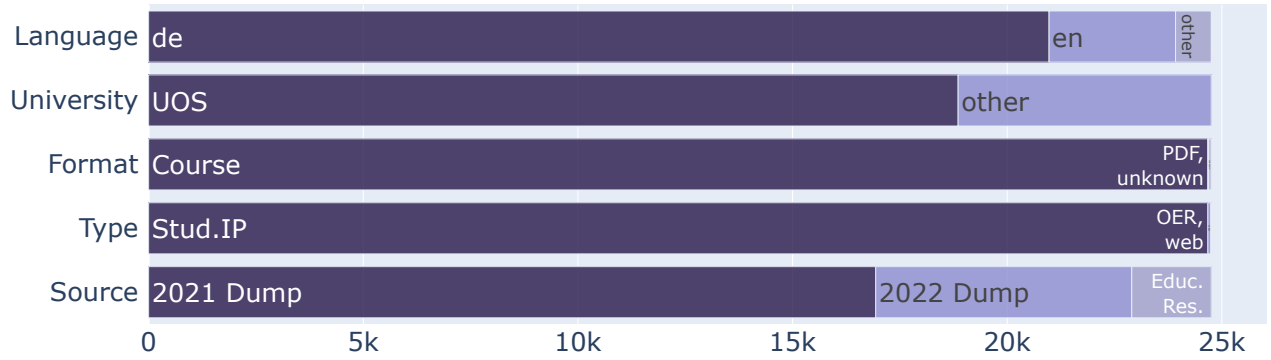

Figure 3.1.: Distribution of metadata in the raw Siddata-dataset. Languages are reported as detected (see B.1), other metadata as it was available in the dumps.

Resource Content

Considering that most of the data are Stud.IP courses, in the following only this format will be described.[15] The most relevant properties for the cause in this thesis are a courses' title, possibly its event number, and its description. The description consists of whatever the creator of the course wrote to describe it to aspiring participants. These descriptions differ a lot in their informativeness, and importantly they are no concatenation of reviews or similar from multiple persons but a continous text description. The distribution of the number of words per description is depicted in Figure 3.2, showing that 98% of descriptions consist of maximally 339 words.

Duplicates There are duplicates in the dataset: Often, the same course is offered over multiple years, which is mapped on to separate resources in the raw dataset. Unfortunately, there is no bijective mapping of course titles or event numbers, meaning that the same event number sometimes refers to the same course over the span of multiple years.[16] The name of courses also may change throughout the years it is offered. A very

[14] But the raw data is available for members of the SIDDATA-project at `https://git.siddata.de/jschrumpf/study_behavior_analysis` (requires authentification), and the dataset-creation notebooks for this dataset can be found at `https://github.com/cstenkamp/MAAnalysisNotebooks/tree/main/create_datasets/siddata`

[15] As the final dataset will only consider resoures with a text-length of at least 80 words, the proportionality of courses is even more pronounced (see Table 3.4)

[16] see Table C.1

Source	Type	Format	Uni	Lang.	≥20	≥50	≥200	≥500
2021 Dump	**Stud.IP**	**Course**	**UOS**	**de**	18203	14551	2671	100
				en	1950	1773	616	53
				other	681	612	100	1
2022 Dump	**Stud.IP**	**Course**	**UOS**	**de**	1009	807	188	5
				en	132	123	70	13
				other	20	14	1	-
			other	**de**	5246	4353	1246	53
				en	771	622	260	25
				other	63	46	9	2
Educational Resources	**OER**	**PDF**	**other**	**de**	51	22	2	-
				en	1	-	-	-
		unknown	**other**	**de**	125	50	6	-
				en	7	4	-	-
	Stud.IP	**Course**	**UOS**	**de**	993	684	121	5
				en	236	163	53	2
				other	168	130	91	25
			other	**de**	737	514	145	23
				en	303	258	89	23
				other	33	23	3	-
	Udemy-MOOC	**Course**	**other**	**en**	22	-	-	-
				other	3	-	-	-
		MOOC	**other**	**en**	8	-	-	-
	web	**unknown**	**other**	**de**	48	3	-	-
				en	1	-	-	-
				sum	30811	24752	5671	330

Table 3.4.: Metadata of the Siddata-dataset. Languages are reported as detected (see B.1), other metadata as it was available in the dumps. Column titles encode the number of entities whose description has at least 20, 50, 200 or 500 words.

prominent characteristic is for example that courses cancelled due to the COVID-19-Pandemic prepended an information such as !!FÄLLT AUS!! to its title, or that a few sentences explaining that the new iteration will be performed digitally. Some techniques have been used to merge possible duplicates based on their title, number and description, however not all duplicates could be eliminated. [17] If the same title was connected to different descriptions in the dataset, the sentences of all duplicating samples were tokenised, concatenated and duplicates removed, such that the resulting description contained every sentence of the original description exactly once. This means that in cases where sentences were only slightly changed, both variants will be present. This is possible because the algorithm only relies on the Bag-of-n-grams representation for these texts (Section 2.5.1), which is weighted and ignores word order.

Possible Classification Targets To evaluate the usefulness of our algorithm to recommend resources from this dataset, we will test how good the extracted semantic directions predict human classes extracted from the data. For that we need classification targets

[17] Many such duplicates were mapped towards similar embeddings by the algorithm, see Section 4.3.

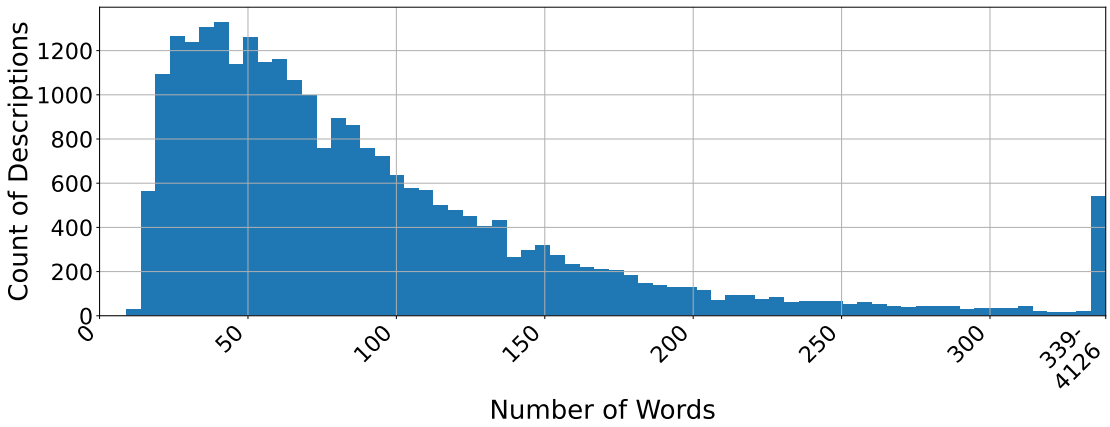

Figure 3.2.: Distribution of description-lengths of the Siddata-dataset. The rightmost bar represents the longest 2% of descriptions.

that can be extracted from the data. As stated above, those resources that are courses from the university of Osnabrück also have their course number given. The first digit of that encodes the faculty a course belongs to, making the faculty an easily obtainable classification target for 75% of the entities.[18] The distribution of the faculties is depicted in 3.3. If the dataset quality should turn out to be very low, it would also be possible to use a classifier that predicts the faculty from its description, and only use those ones that were correctly classified by it, which theoretically would ensure that only courses whose description is meaningful would be obtained.

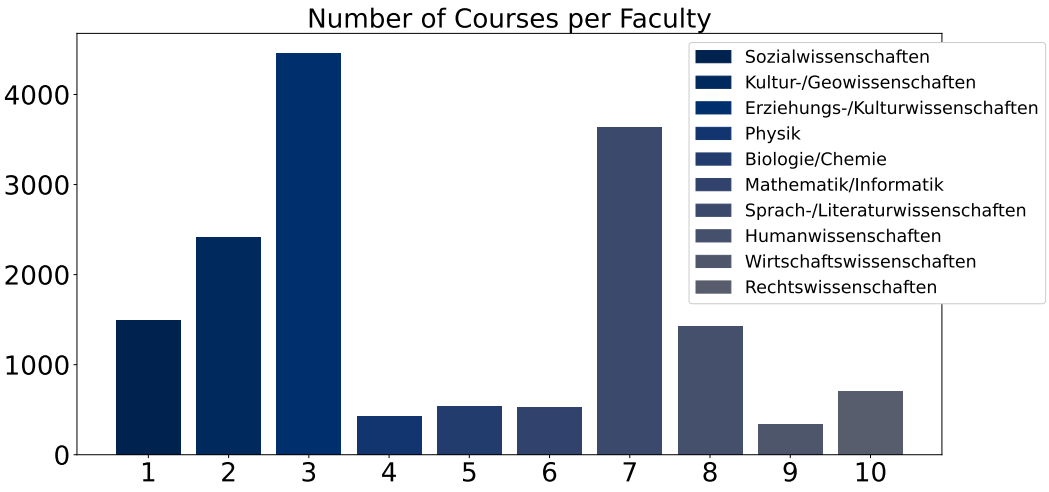

Figure 3.3.: Distribution of faculties for those courses at University of Osnabrück.

Other Metadata Apart from *title*, *description* and *course number*, the individual resources may have some additional properties. Among these are *format*, *type* and *source* as meta-information encoding their origin and type. As most of the resources come from an older dump, these are not given for all. A small fraction of rerources additionally lists *subjects*, which is a list of keywords describing the course. This is very relevant information for the algorithm, however as only a tiny fraction of entities lists it, the information is

[18] Which is a lot more than there are for the placetypes-dataset, see Figure C.1

just appended to the respective description, just as the *subtitle* of a course. Next to some other information only given for resources that are not of the format *Course*, the only abundantly available metadata is the DDC-Code that was assigned to it by SidBERT (see Subsection 2.4), which may be used as additional classification-target.[19] Besides this, there are unfortunately no further obvious candidates to generate prediction-targets from. This problem was discussed with the author's supevisors, however the process was complicated due to privacy concerns, anonymization and inappropriate data formats.[20]

Implications There are quite a few duplicates in the dataset, the descriptions are shorter and different than the original ones, and many descriptions seem to be short and uninformative. The low dataset quality is a reason to later look at our results with a grain of salt: If for examply quantitatively examining if different courses are mapped onto the same embedding, it must also be qualitatively looked at these samples, as it may be the case that they are in fact multiple entries of the same course and thus correctly detected duplicates.

Low Quality Samples Unfortunately, the dataset contains many samples that have only very short and uninformative descriptions. The analysis-script in the repository[21] does some exploration regarding length and quality of the data. Low-quality descriptions include: *"BA/MA Hauptmodul" - "Bestandteile:Vorlesung + Übung" - "Dozent Dr. Michael Wicke" - "Siehe Gruppe A" - "s. Modulbeschreibung" - "Literatur:wird noch bekannt gegeben"*

3.1.2. Placetypes

Because the Siddata-dataset differs in many regards from the datasets used in [1–3], it makes sense to compare the results of the given pipeline on that dataset to the results of [1–3]. To be able to compare the results achieved here with theirs, one of the datasets used by all of their papers was considered as well. Comparing on an original dataset also has the benefit of allowing to sanity-check if the implementation is correct: If the performance achieved on this dataset is comparable to the performances of [1–3] whereas the peformance for the Siddata-dataset is considerably worse, there is strong indication that the quality or quantity of the dataset is insufficient for acheiving high-quality representations. Conversely, if our algorithm applied to the dataset used in [1–3] produces results that fall considerably short of those reported in literature, it is likely that there is a fault in our implementation.

There are two datasets used by all three authors: placetypes and movie-reviews (see Table 3.3). Both of the datasets are available in preprocessed form[22] [1]. It was originally planned to use both of the datasets as basis for comparison, however unfortunately it is impossible to recover the form of it as originally used by Derrac & Schockaert [1]: The

[19] For that, however, one has to keep in mind that these were also only produced by a ML-algorithm which has no perfect accuracy.

[20] One plan was to get the semester of a courses' enrolled students, but this data is stored timeless, and the mapping of the participation year is obfuscated through k-anonymization.

[21] `https://github.com/cstenkamp/MAAnalysisNotebooks/blob/main/create_datasets/siddata/data_exploration_Siddata2021.ipynb`

[22] `https://www.cs.cf.ac.uk/semanticspaces/`

original texts of the reviews are not available, but only their respective bag-of-1-grams (BoW) - even though the authors explicitly state that they worked with variable-length-n-grams. Even though the provided list of candidates contains n-grams, it is impossible to recover which of the entities contained it. While the algorithm can still be run only for the 1-grams, the results are not comparable with the original ones anymore.[23]

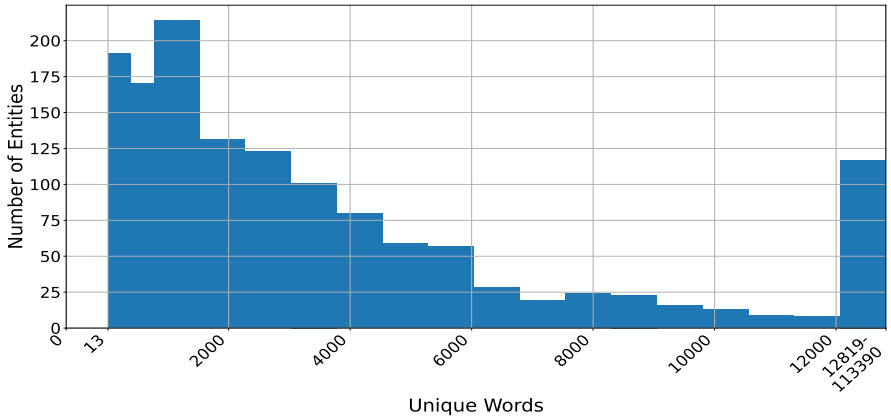

Figure 3.4.: Distribution of unique words per entity for the placetypes dataset.

In this work, we will only compare the results w. r. t. the GeoNames[24] and Foursquare taxonomies, because the level used for the openCYK taxonomy was not sufficiently described by Derrac & Schockaert [1]. For the GeoNames taxonomy, 403 of the placetypes are split into 9 categories. These labels are annotated to a t-SNE plot genreated from Derrac & Schockaert's [1] uploaded data in Figure C.1 in the Appendix. In the case of Foursquare-labels, 391 placetypes are split among 9 (different) categories as well.

The dataset was created from tags of images shared flickr[25]. A picture was considered to be of a certain type, if one of its tags is the name of that type according to one of the aforementioned taxonomies, and the BoW-representation considered for this algorithm was created from all other tags of these pictures. In contrast to our dataset, this one was specifically created by Derrac & Schockaert [1] to test their claims of the algorithm implemented here. Preliminary inspection of the dataset has however revealed that it however still contains many duplicates, such as the entries *boat yard* and *boatyard*, or *skate park* and *skatepark*, among others.

3.2. Algorithm

Let us finally go into detail about the main algorithm. The implementation of this thesis replicates and extends the algorithm proposed by Derrac & Schockaert [1] with some novel contributions to deal with the given dataset. Further, some improvements from the works of Ager *et al.* [2] and Alshaikh *et al.* [3] are incorporated, who also replicated and improved the original algorithm and published together with Prof. Steven Schokaert. According to

[23] As all of the [1–3] share an author, it is assumed that [2, 3] had access to the full version of the dataset and did not share the stated problem.

[24] `http://www.GeoNames.org/export/codes.html`

[25] `https://www.flickr.com/`

our evaluation of the field, these two papers provide some useful improvements in several aspects, as they apply the algorithm to different datasets, suggest more straight-forward ways of evaluating their performance, and help in understanding important concepts. That we are focusing on only those three papers should by no means imply that they are the only ones that were considered and influcenced this implementation,[26] however in contrast to the other pertinent literature these two works do not substantially divert from the algorithm's core principles.

It is important to keep in mind that the algorithm is no rigid monolith but modularly consists of several components, such as *dimensionality reduction*. Many of these components do not require specific algorithms, and [1–3] also experiment with different components. The exact system for these components may be exchanged, and in the following these exchangeable algorithms are also referred to as hyperparameters. Note further that while this thesis mostly replicates the work of Derrac & Schockaert [1], the following will describe the algorithm as implemented here, which differs in some details from the original work. For the sake of clarity, very specific implementation details will be left out in the following description, as however reproducibility is an important aim for us, implementation details are available in Appendix B and linked where relevant.

Core Algorithm

The main goal of the algorithm is to unsupervisedly use text-corpora associated with the considered entities[27] to embed these into a vector-space where the axes correspond to human concepts/properties.[28] This is referred to as *feature-based* representation: A high-dimensional vector that numerically encodes the degree (*protoypicality*) to which the entity corresponds to a number of appropriate dimensions. This is generally referred to as Conceptual Space and can be used as basis for explainable reasoning.

The general idea to achieve that is as follows: First, the entities are embedded as fixed-dimensional vectors. To allow for the types of reasoning from Section 2.3.2, it is embedded into metric spaces where the concepts of direction and distance are well-defined. Derrac & Schockaert's [1] original algorithm uses MDS (see 2.5.1) for this matter, which enforces metric distances. [2, 3] both soften this requirement and also use document embedding techniques such as Doc2Vec and averaged GloVe [50] embeddings. Additionally, words or phrases from the text are extracted as candidates for the names of the semantic dimensions. The underlying assumption is that "words describing semantically meaningful features can be identified by learning for each candidate word w a linear classifier which separates the embeddings of entities that have w in their description from the others" [3, p. 3574]: The better the performance of that classifier according to a chosen metric, the more evidence there is that w describes a semantically meaningful feature. In a final step, the candidate-words are clustered according to their similarity to find a fixed set of *semantic directions*. A representative term for the directions is selected as dimension name,

[26] See Section 2.4

[27] From now on, the term *entities* refers to the sample described by one text from the corpus (description, concatenated reviews, ...) from a certain domain. The corpus accordingly defines the domain: educational resources, movies, ...

[28] *Concepts* and *Properties* explicitly refer to what is defined in Criterions C and P, see 2.3

and the entities are re-embedded into a new space comprised of these dimensions, where the individual vector-components correspond to the ranking of an entity with respect to these dimensions.

The rest of this section goes into further detail for each of the individual algorithm components. Further, configuration files to enable exactly the respective components of the papers [1–3] for the codebase of this thesis are listed in Appendix B.3.

3.2.1. Algorithm Steps

This section describes the steps how to create an interpretable vector-space from the text corpus in detail. For that, we will explicitly elaborate on the parameter choices that branch up at every step for this specific implementation.

1. **Preprocess** the corpus with default techniques and create a *Bag-of-ngrams representation* (2.5.1) of the texts.
2. **Extract Candidate Feature Names** from words/n-grams of the corpus.
3. **Embed all Entities** into a fixed-dimensional vector space with demanded properties that captures the respective semantics.
4. **Filter Candiate Features** by training a linear classifier for each candidate that seperates the vector representations of the entities that contain the term from those that do not. If a specified metric for this classifier is sufficiently high, assume that the candidate term captures a *salient* feature - its direction is then characterized by the orthogonal of the classifier's separatrix.
5. **Cluster/Merge the candidates** and calculate the feature direction for each cluster from its components, and (optionally) find a representative cluster-name.
6. (optionally) **Post-process** the candidate-clusters.
7. **Re-embed the entities** into a space of semantic directions by calculating their distance to each of the feature direction separatrices.

This techniques first embeds the collection of texts into a vector space, to afterwards extract important features from this space using linear classifiers. The second step is an original idea of [1], however creating vector space embeddings from texts is a very popular technique, used for many tasks in NLP [12, 13, 39, 41, 43]. This implementation relies on classical creation of the VSM, for which the general creation process was explained in Section 2.5.1. The steps *Build the Frequency Matrix*, *Transform Raw Frequency Counts* and *Smooth the Frequency Matrix* are squashed into the preprocessing and embedding of entities. When *extracing candidate features*, their frequencies must additionally be quantified - which may differ from the quantification when *embedding all entities*.

An explicit and simple implementation compliant with each step could be a simple word tokenization and count to generate a bag-of-words (step 1) where each sufficiently frequent word is used as candidate (step 2). A dissimilarity matrix of the individual BoW-vectors is compressed using MDS (step 3). A SVM calculates the accuracy for each candidate (step 4), and k-means-clustering on the 500 top-scoring terms subsequently creates 100 clusters and averages their directions (step 5). The distance to each of the hyperplanes is calculated (step 6), yielding new space for the entities. The sequence of steps is also given as pseudocode in Appendix D.

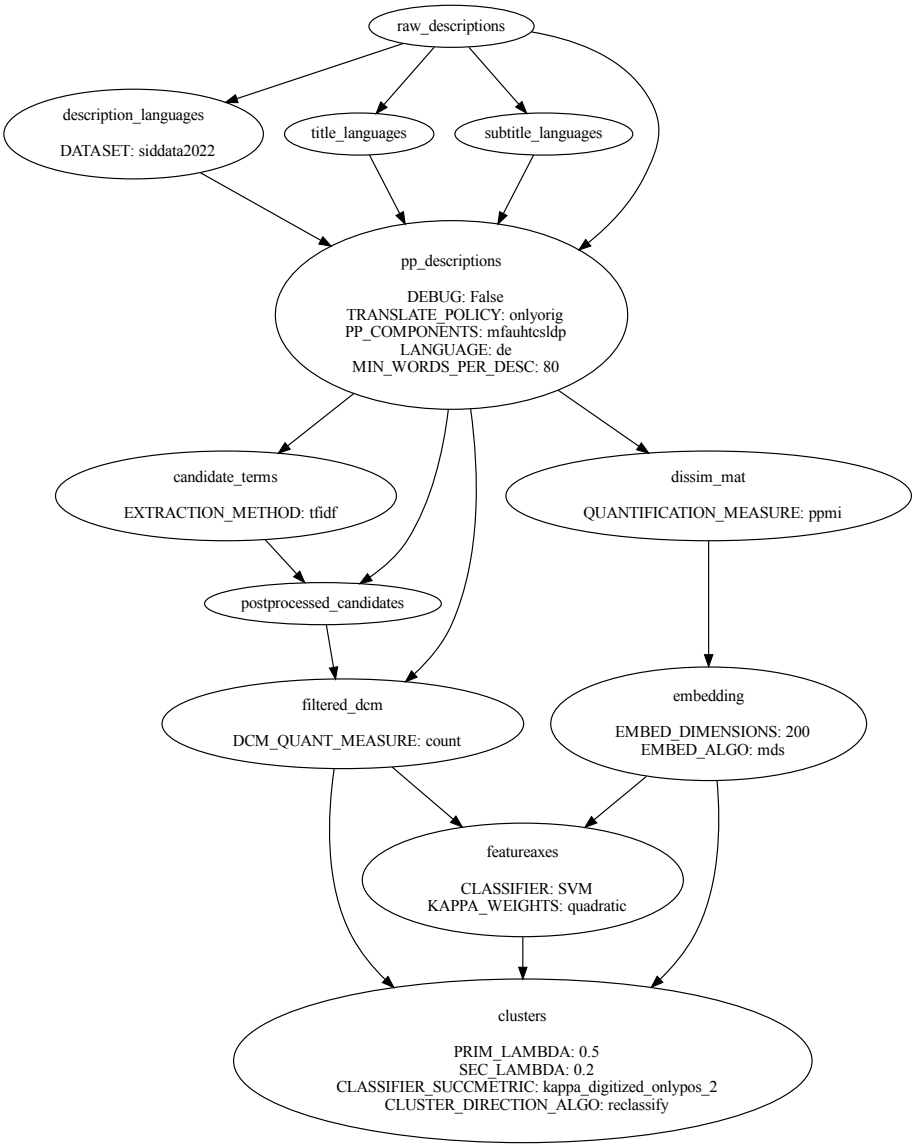

Figure 3.5.: Dependency-graph of the implementation, displaying the individual steps of the algorithm as well as their dependencies and where selected important parameters are first used.

The distinction of steps is not always this rigid: Instead of creating a dissimilarity matrix followed by dimensionality reduction, [2, 3] use neural word- or document embeddings. We will come back to other ideas ideas when discussing future research opportunities (Section 5.4) by listing what other ways of fulfilling each respective step could have been considered.

Figure 3.5 shows an automatically exported dependency-graph, displaying the individual steps of the algorithm as done in the accompaning code, also showing where selected important parameters are first used. As explained in Section 3.3, the modularity of the

provided architecture allows individual components to be exchanged as needed and run in parallel.

Preprocessing

Before looking at the steps in turn, it should be noted that even the preprocessing does not work on completely raw data, but on curated and processed corpora. This processing is however not considered part of the algorithm, as it is very specific to the respective datasets and manual dataset exploration, tweaking settings such that they are best for each corpus separately. The preprocessing for the Siddata-dataset is described in Section 3.1.1 and its implementation is done in separate Jupyter Notebooks.[29] In the considered literature, the preprocessing is not considered part of the algorithm at all. Their implementations start from already fully processed datasets available as bag-of-words, each separately processed. Details of their individual processing per dataset is listed in Table 3.3. By incorporating the preprocessing into the pipeline, this work aims to increase adaptability and reproducibility, and also allows to experiment with different techniques such as translation or lemmatization or how duplicate entities with different associated texts are merged.

A common prerequisit for NLP algorithms is to pre-process the text corpus. The pre-processing itself consists of multiple independent components chained after each other. Which components are necessary also depends on the processed dataset - as for example the *placetypes*-dataset consists of a collection of *tags* instead of full sentences, tokenising sentences or removing stop words becomes irrelevant. Other datasets may require additional cleaning or are already available in preprocessed form.

Translation As the main considered dataset of university-courses is highly multilingual (see Figure 3.1), one of the first questions that needs to be addressed is how entities of different languages are handled. The algorithm consists of classical language processing algorithms such as comparing BoW representation of the entities, which means that the same text in two different languages may result in maximally different representations (see Section 2.5.1). Because of this, before any other processing, the languages of each entity is checked, such that those of languages other than the demanded may be either translated, left out or used anyway. For details about the translation, it is referred to Appendix B.1.[30]

Components The following components are developed for the preprocessing, every one of which can be individually enabled or disabled:

- Prepend title and/or subtitle to the entities' associated text
  useful for the Siddata-Dataset, as the titles are often quite long and more descriptive than their descriptions
- Remove HTML-Tags from texts
  useful for the Siddata-dataset, as it includes descriptions for MOOCs which are crawled from websites and often contain such

[29] Such as `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/notebooks/create_datasets/Preprocess_Siddata2022.ipynb`

[30] It should be noted that professional automatic translation is costly and thus not all texts are available in all languages.

- Tokenise sentences

  such that n-grams across sentences are not considered
- Lower-case all words

  reduces the amount of individual words and ensures that words at the beginning of sentences are mapped correctly
- Remove stop-words / frequent phrases
- Tokenise words

  means splitting at the word-boundary, resulting in a list of words. Order must be kept in case n-grams are to be extracted.
- Lemmatize words
- Remove diacritics

  Diacritics *are glyphs added to basic letters, such as accents or German* Umlaute. *Removing them converts for example the letter* ä *to an* a
- Remove punctuation

The above can be done either be done with proprietary code for all of these steps,[31] or using `sklearn`[32] `CountVectorizer` (which is faster, but less configurable), as [2] claim to have done.

On Stop-Words Removing stop-words from the texts is useful because it makes the resulting frequency more compact and thus less computationally intensive, and stop-words generally have very discriminative power, meaning their occurence among the entities is arbitrary, just making hte emeddings more noisy (cf. Section 2.5.1). There are however reasons to not remove them: Two words that are considered stop-words may posess relevant semantic content (such as a FÄLLT AUS in a course title), and stopwordslists are often incomplete and of low quality [51]. For these reasons it is also possible to instead remove n-grams that exceeded a certain frequency (df).

On Lemmatization The languages most prevalent in the considered datasets are considered *agglomerative*, which means word stems are changed by the addition of affixes and suffixes. Consequently, the same word may be present in multiple different forms, which modelled as completely dissimilar vectors in the present BoWs-approach. Lemmatization is the process of mapping different forms of these words onto the same stem. Considering that the Siddata-dataset consists of far fewer words than the others, this has important implications. For the german descriptions, this implementation relies on the *HanTa* lemmatizer [52].[33]

The result of this step is a bag-of-ngrams representation for each entity (see Section 2.5.1).

Extract Candidates

The final result of the algorithm is a metric space in which the individual dimensions (*features/interpretable directions*) correspond to natural-language concepts and attributes. The candidates for these features are verbatim phrases extracted from the text-corpus of the entities, which are subsequently filtered and merged as necessary.

[31] Mostly relying on the python package `nltk` [42]
[32] `https://scikit-learn.org/stable/`
[33] `https://textmining.wp.hs-hannover.de/Preprocessing.html#Lemmatisierung`

In Derrac & Schockaert's [1] work, the selection of phrases to be extracted depends on the dataset: For placetypes-dataset, all sufficiently frequent[34] 1-grams[35] were considered. For the other two datasets, they applied a POS-tagger that extracted all sufficiently frequent[34] **adjectives, nouns, adjective phrases** and **noun phrases**, assuming that adjectives would correspond to gradual properties (e. g. *violent, funny*) and nouns to topics (e. g. the *genre*) [1, Sec. 4.2.1]. Also, the authors ensured that the number of extracted candidates for both datasets is roughly equal, getting around 20 000 candidates for movies and placetypes.

For this step, the implementation of this thesis differs from the original algorithm, as both taking all words as candidate and running a POS-tagger led to suboptimal results in previous experiments, which indicated that the robustness of the algorithm is increased if less candidates are considered in earlier steps. This will be further argued and elaborated in the discussion. To ensure comparability to these works however, in the case of the placetypes-dataset the original method of taking all words with a term-frequency of at least 50 was used. Similar techniques for the Siddata-dataset were also considered, but in constrast to the placetypes-dataset it is also important to consider various-length n-grams.

In our implementation the candidate-extraction is split into four subsequently excecuted substeps, because depending on the algorithm used to extract the candidates the runtime of the individual components is comparably long and some settings are only relevant in later substeps. The steps are:

- Extracting Candidate Terms
- Postprocessing the Candidates
- Creating the document-term matrix for the candidates and applying a quantification

As visualised in Figure 3.5, these substeps only depend on the preprocessed descriptions, which means they can be run in parallel to the creation of the embedding if e. g. running on compute clusters.

Three main techniques are implemented to extract candidates from the text-corpus. Irrespective of the algorithm, only words with a sufficiently high df are extracted, which is important to ensure that the classifier that determines its meaningfulness has enough samples in both clases. This means that the minimal freqeuncy can be calculated from the dataset size: In [1], the minimal frequency for the movies-dataset with 15 000 entities was only 100, meaning that the algorithm even works if only 0.6% of samples are in the positive class.

By frequency: consider all phrases that exceed a specified document-frequency (like [1]).

By a quantificiation: consider all phrases that are prominent by some notion of *importance* , such as the PPMI or tf-idf-score. Note that the respective scores depend on the combination of document and term, such that candidates may be extracted for

[34] The respective thresholds are listed in Table 3.3 as "candidate word threshold".

[35] Note that in the case of the placetypes dataset, a 1-gram corresponds to all merged words of a tag.

some documents. Of course, all their occurences are considered in the creation of the frequncy matrix.

Using *KeyBERT* [53]: consider phrases whose BERT-embedding [15] is most similar to the text they are in.

Using KeyBERT results in candidate terms that are most appealing in qualitative inspection, however it is also most computational demanding, techniques and requires substantial amounts of post-processing for the resulting phrases. More details on KeyBERT and how it is incorporated into the algorithm are given in the implementation are given in Appendix B.1

Finally, a document-term matrix is created from the postprocessed candidates, containing the frequency for each candidate-phrase in each entity. The creation of this frequency matrix mirrors the process described in Section 2.5.1, however only for the extracted words. After filtering this matrix to ensure that only candidates with a minimal df or *stf* are considered, a quantification is applied as described in Section 2.5.1. Available Quantifications include raw count, binarization[36], tf-idf or PPMI. We expect that expressing the relation of terms and documents by something else than the raw count will prove especailly useful for the Siddata-dataset: As it does not consist of concatenated reviews but short descriptions, each individual word will only occur a few times per description. Because of that, a rank induced from the count will be not very meaningful when later comparing it to the ranking induced by the classifier.

Generating Vector Space Embeddings

In this step, the individual entities are embedded into a fixed-dimensional vector space, making up a *frequency matrix*.[37] Neither this nor the frequency matrix from the previous step will used to finally calculate similarities on, but both are interim results to get the dimensions necessary for these similarities.

Embedding words, n-grams/phrases or other tokens, as depicted by [39, 41], generally involves counting the token frequencies, transforming them to get relative frequencies, and performing dimensionality reduction on the resulting matrix.

So far (step 1), we have counted the token frequencies. Derrac & Schockaert [1] argued that this space must be Euclidean, such that geometric/algebraic solutions correspond to commonsense reasoning tasks (see Section 2.3.2). Another requirement is that the number of dimensions is be chosen as hyperparameter to the algorithm to be able to find a compromise between compression and expressive power. Because of these two requirements, they selected MDS for dimensionality reduction.

As stated in Section 2.5.1, MDS calculcates a Euclidean VSM from a set of pairwise distances. This means that the algorithm first creates a *Dissimilarity Matrix* that encodes the distance between all pairs of entities (represented as Bag-of-ngrams representation), from which subsequently the final embedding is generated.

[36] When using binarization, all counts are either one or zero. According to Alshaikh *et al.* [3] this improves performance, however we could not confirm this.

[37] The previous step already created a frequency matrix that encodes the relevance of a candidate-phrase for each entity- this is a spearate one that later serves to encode each document as a vector

In this implementation, the dissimilarity matrix is created using distance metrics for the quantified bags-of-ngrams of the respective entities. Note that this quantifiation does not need to be the same one as the one used for quantifying the relatedness of the extracted candidates and the documents.[38]

Document embeddings If the strict requirement for a metric space is dropped however, many different algorithms may instead be used at this point - not only different dimensionality reduction methods for the embedding, but also ones that do not rely on the distance matrix or even the BoW at all, like document-embedding-techniques such as Doc2Vec [14] (as e. g. used by [3]). This would not require the count of token frequencies followed by a quantificiation, but not affect the rest of the algorithm. As, however, document embeddings tend to be created on the basis of cosine distance, this will not result in a Euclidean metric. Alshaikh *et al.* [3] experimented with this, however it performed worse than the original algorithm relying on MDS (see Table C.2), which is why this technique was not futher pursued in this work.

Classical Processing The default way of doing it is, however, to create frequency matrix counting the token frequencies of each word of an entity and using a quantificiation to transform these raw counts. In this implementation, the dissimilarity matrix that is required for the MDS-algorithm created on basis of the *normalized angular distances* of the BoWs of the respective entities, which is relates to the cosine distance and is defined by Derrac & Schockaert [1] as:

$$\begin{aligned} ang(e_i, e_j) &= \frac{2}{\pi} * \arccos\left(\frac{\mathbf{v_{e_i}} * \mathbf{v_{e_j}}}{\|\mathbf{v_{e_i}}\| * \|\mathbf{v_{e_j}}\|}\right) \\ &= \frac{2}{\pi} * \arccos(1 - \cos(\mathbf{v_{e_i}}, \mathbf{v_{e_j}})) \end{aligned} \tag{3.1}$$

Finally, the dimensionality of the frequency matrix is reduced using the MDS-algorithm described in Section 2.5.1. [1] also experiment with the *Isomap* algorithm, which does not yield a Euclidean space but one of geodesic distances. As the results for their implementation are, however, of worse quality than those for MDS, it is not considered further in this work.

In the publication of Derrac & Schockaert [1], the result of this embedding is used to experiment with the explainable classifiers (discussed in Section 2.3.2). Due to the usage of MDS, this space has clearly defined concepts of *betweeness* and *parallesism* between the entities, however it is important to stress that it does not have interpretable dimensions and is, so far, a plain VSM as described in Section 2.5.1.

Filter Candidates by Classifier Performance

This step brings together the entity embeddings and the extracted keyphrases, to finally find semantically meaningful directions in the space the entities are embedded in. For that, classifiers are trained for all previously extracted candidates, each yielding a *Candidate Feature Direction*, which are subsequently filtered to detect those corresponding to semantically meaningful features. The algorithm is executed separately for each candidate term and looks as follows:

[38] As will be shown in the result, doing so sometimes even outperformed other configurations.

All entities are split into two classes: Those that contain the respectively handled candidate term and those that do not. To quantify how well each candidate word captures semantic content of the entities, a linear classifier is trained to separate the embeddings of these two classes. The best example for such a classifier is a SVM which does not rely on the kernel trick. As visually exemplified in Figure B.1, the result is a hyperplane that best divides the positive from the negative samples. Regardless of the dimensionality of the original space, this hyperplane has a one-dimensional orthogonal vector. Each of the entity-embeddings is subsequently orthogonally projected onto it. The distance of this projection to the plane offset[39] is a scalar that encodes the distance to this decision hyperplane.

This distance encodes *how much* this point is considered to be protoypical of the respective class by the classifier, for the positive as well as the negative class. Given that the original space is created based on similarity measures of the entities, the *Bag-Words-Hypothesis* (Subsection 2.5.1) states that similar words should have similar words - meaning maximally dissimilar entities are far apart from positive ones.[40] Because of this, the hyperplane's orthogonal can be considered an axis that encodes how much each entity agrees with the feature that is the basis of the classification. Accordingly, a ranking of the entities in terms of this distance should encode their degree of having the respective feature.

According to Derrac & Schockaert [1], the performance of this classification encodes how good a candidate serves as feature direction. This can be explained like this: As discussed earlier (Section 2.5.1), unimportant words are arbitrarily throughout the corpus. Due to distibutional semantics, topics that are very prominent in some texts but not in others will influence the position of a texts' embedding in the vector space more than *unimportant* words, which do not signify a latent topic of the corpus. This means, they do not correlate with other words and do not explain much of the dissimilarity of the embeddings. Accordingly, unimportant words do not influence the positions of the entities' embedding other than being *noise*. This randomness does not go along with a cluster of positions in any of the dimensions, as noise gets removed in process of dimensionality reduction. This makes it reasonable to assume that the a classifier can split entities that contain a word from those that do not, the more the word is an *important topic* in the sense that it explains the dissimilarity in the entities.

Because of this, [1] henceforth only consider those terms as *faithful directions* that explain a lot for variance in the data whose performance exceeds a certain threshold.[41]

Concretely, the score used by [1] to assess the performance is not the accuracy or some other measure of the binary performance of the classifer, but rather if the ranking induced

[39] The *offset* encodes the coordinate where the projection crosses the decision surface.

[40] To stick with the example of movies, the assumption is that movies that are maximally unscary are maximally far from the away from scary ones: An entity that has a maximally dissimilar distribution of words than those that are *scary* means a maximally maximally dissimilar movie, as little of its *latent topics* (see Section 2.5.1) agrees with the scary ones: The more dissimilar, the less scary.

[41] "if this classifier is sufficiently accurate, it must mean that whether word w relates to object o (i.e. whether it is used in the description of o) is important enough to affect the semantic space representation of o. In such a case, it seems reasonable to assume that w describes an important feature for the given domain." [2]

by the classifier corresponds to ranking of number of occurences[42] of that word: The more these agree, the more we consider this direction "to be a faithful representation of the term" [1, p. 20]. The reasoning behind that becomes especially clear when considering the root of their datasets - in the case of reviews or tags it is the case that the more often a word is mentioned, the more relevant the word is for that entity. In the case of using a quantificiation such as the PPMI-score, this instead becomes: The more salient relevant for this entity but not for the others the word is, the higher the score.

A score function compares these rankings, such that only those terms where the correspondance of these rankings exceeds a certain threshold are considered as candidate directions henceforth. For that, Derrac & Schockaert [1] say that they use the Cohen's kappa, which is a metric to compare rankings that can deal with highly imbalanced data. Considering that for most candidates, most of the entities will be in the negative class this makes sense.[43] Unfortunately, the authors do not give many details on how this scoring is implemented. While [2, 3] explicitly say that they are interested in the PPMI-scores[44], from [1] it is not even clear if they take the count or the PPMI-score. As that is highly relevant, we try many different ways of this scoring and report them in the results. We also compare the overlaps of different kappa-scores to check if the choice is as imporant as we think it is. Which scores we used and how they are written here is listed in the implementation details: Table B.1.

Subsequently, only those candidates with high enough scores are considered, and the othogonals of the respective classifiers considered their respective *candidate feature direction*, encoding the degree how much an entity corresponds to this feature. The number of features with a sufficiently high score can also serve as estimate of how good the parameter-combination so far was: if not enough well-scoring features were extracted, the final embedding which is created through only a subset of these features does not explain much of the original variability in the data.

Merging the extracted candidate-directions

The previous step yielded many *basic feature directions* that are defined as direction of the orthogonal vector for the hyperplanes splitting each individual candidate n-gram. The performance-thresholds are set such that many more directions are generated than the demanded dimansionality of the final embedding, such that they must be clustered and merged.

This is done via the following substeps, each of whch will be closer eloaborated:

- Cluster good-performing candidates by their similarity
- (optional) Remove uninformative features
- Recalulate the direction of the cluster
- (optional) Find a representative name for the cluster

[42] or the PPMI-score, the authers are imprecise in their wording - we will elaborate more on that later

[43] Ager *et al.* [2] compare the kappa-score to accuracy and argue that accuracy works better than the kappa-scores.

[44] Though the uploaded code of [3] does not compare rankings but raw values.

Clustering the candidates Clustering refers to an unsupervised algorithm that groups items based on some notion of similarity. In our case the assumption is that semantically similar concepts have *close* vectors, which is given due to the BoW-hypothesis that states that the underlying structure of our dataset is expressed by the usage of related words (Subsection 2.5.1, 2.5.1).[45] As these vectors in principle only encode a direction, their similarity can be calculated by their cosine distance.

The clustering should reduce the number of features and also ensure that the resulting directions are different enough. Note that unlike e. g. in Principal Component Analysis (PCA), the suggested here techniques do not enforce orthogonality, such that the resulting directions may remain linearly dependent to a certain degree. As in the final embedding only the projection onto those directions is relevant, it must be ensured that enough of the data's original variation is covered by these directions. To ensure that, we follow Derrac & Schockaert's [1] suggestion to allow for redundancy by extracting twice as many directions than the original VSM dimensionality.

The following steps describe the implementation of the original clustering method of Derrac & Schockaert [1], also used in our work:

First, we consider the best *basic features* as *main directions*. For that, we select one of the scores calculated in the previous step and select all candidates that exceed a threshold ([1] suggest $\kappa \geq 0.5$). To get the directions, we follow the following algorithm:

```
greats = filter(candidates, 0.5)
directions = greats.argmax()
for nterm in range(ndims*2):
  greats = set(greats)-set(directions)
  distances = {cand: min(comparer(cand, compareto)
                for compareto in directions)
                  for cand in greats}
directions.append(compares.argmax)
```

This starts with the best candidate and then iteratively adds the one from the set of top-scoring candidates that most dissimilar to the set of final directions. The result is a set of $ndims * 2$ main directions, which are henceforth considered the Cluster centers. Subsequently, all leftover terms from $T^{0.5}$ as well as all terms from $T^{0.5}$ are added to the respective cluster whose direction they are most similar to.

Derrac & Schockaert [1] used the cosine distance to measure the respective similarities. This may lead to unexpected situations (discussed in Subsection 5.2.2). As alternative similarity metric that does not rely on the angle between their vectors, [35] suggest to use the overlap of the positive-samples of two features as similarity. This was however not yet implemented in this thesis.

Alternatively, to the described algorithm, is also possible to use the popular *k-means* algorithm for clustering, as done by [2]. We do not present results for this approach here however, as it lead to a substantial increase in runtime, without affecting performance much. In the development we also noticed that many clusters contain a lot of irrelevant terms. To alleviate this, we experimented with different techniques, for example setting

[45] In case of the Siddata-dataset, it may mean that in courses that contain the word *computer* have a high chance of also containing *program*.

minimal similarity thershold that must be given for a term to be added to a cluster, however so far no formal evaluation to test how this affects performance was performed.

Find Cluster-Direction So far, we have a set of clustered canidates terms, each of which has an individual direction. The final *feature-direction* must subsequently be found from the elements of the cluster. For that, [1] and [2] define the cluster centroids as the average of all (normalized) vectors per cluster. In our experiments, however, we noticed that the final direction tends to be too much affected by irrelevant cluster-elements. Because of this, we experimented with other techniques to determine the cluster direction. Other considered methods include e. g. to 1. just consider the direction of the cluster-center, or 2. to weight the influcence of each cluster-element by their kappa-score.

The best performaning method however was the *reclassify*-algorithm, which (similar to [3]) finds the cluster-direction by training a new classifier that splits those entities that contain *any of the elements* from the cluster from those that do not, analogous to the previous step.[46]. Doing this however often leads to the opposite problem than the previous step, namely that for many clusters there are almost no entities that do not contain at least one of the cluster elements. To counter this, we instead trained a classifier to split the 30% of entities with the highest quantificiations from the 30% of entities with the lowest quantificiations. A comparison of this algorithm with the method of [1] is given in the Appendix as Table C.3. As [3] already performed formal experiments with this that have shown its superior performance, all generated results of this work rely on this algorithm.

Bad Clusters After these steps, we finally have the vectors that correspond to semantic directions. However, as there were still clusters of uninformative terms, Alshaikh *et al.* [3] have an additional step to remove uninformative clusters. As this bases on another clustering algorithm used by the author (*affinity propagation*) which does not allow to specify the number of clusters, it was not implemented here.

Find a representative Cluster Name An important advantage of the clustering process is that it makes the extracted directions more *descriptive* due to them being described by several phrases instead of only one. However, it may be helpful for an attractive user interface to find the single *best* description of the cluster direction by its element.

An analysis of [54] showed that a statistical method to extract features from clustered text corpora identified the labels of human annotators as one of the top five most important terms in only 15% of cases, implying "that human labels are not necessarily significant from a statistical perspective" [54, p. 139]. In their paper, they suggest several methods to find one representative name for the cluster.

Derrac & Schockaert [1] and its follow-ups [2, 3] did not care about such methods and instead use either the name of the cluster center as its description or the cluster center plus two other sample elements. This work experimented with several techniques to get a more representative direction name. One of these techniques used the KeyBERT-algorithm (see Section B.1) to find the term that is most similar to the set of terms making

[46] except that it requires to generate and quantify a new frequency matrix from the sums of the individual counts.

up the cluster. We also experimented with a method that embeds the cluster terms using Word2Vec and returns the word behind the vector that is closest to their average, which is not neccessarily part of the original set of words. Similarly to LSA (Section 2.5.1), it is also possible to consider the entity whose *pseudo-document* embeddings is closest in direction to the cluster direction.

The best technique to find a cluster-name was not evaluated yet. All considered methods that formally evaluate the corresponding feature-directions work independently of the actual cluster name. This is unfortunate, because *subjectively*, the name of the respective directions is very important for the usability of any recommendation engine based on this work. Especially this subjectivity indicates that the only way to evaluate the cluster-names is with a study of human subjects.

Postprocessing the Feature-Directions

The main contribution of Ager *et al.* [2] is a postprocessing step that changes the final space such that the ranking of entities w. r. t. each feature direction more closely mimics the ranking of frequencies of that direction's cluster words. The reasoning behind this is that the original embeddings from which the feature directions are created are based on global similarity. This makes it very vulnerable to outliers which often take up extreme positions. If one now creates the feature directions from the space, these outliers are assumed to have certain properties. The space is optimized for that, which limits the quality of feature directions in the space. The problem here is again the global similarity: If one entity ranks high for a feature, it is very likely that another entity that is close to that will also rank high for this feature, even though it may be something completely different. So to get better feature directions one has to distort the space. The authors accordingly again use the BoW representation of the entites to fine-tune the positions of the embeddings in the final space: After the clusters are collected and the entities re-embedded, for each feature a new ranking is computed by the summed frequency of any of a cluster's words per feature and entity. Each entity is thus represented as Bag-of-Clusters and again scored with PPMI to generate a ranking for each cluser/direction. This ranking is then used as a target for a simple ANN that distorts the space representation. As has been shown that it increases the algorithm performance only slightly while adding a substantial amount of work, it was not implemented in the scope of this thesis.

Re-Embedding the entities into the new space

To finally get the **feature-based representation** of the entities, they are re-embedded into a space where each of the vector components is a semantic directions and the value are the respective rankings. According to the authors, the degrees of similarity are not supposed to be encoded, which is why only the respetive ranking is considered.

3.3. Architecture

As elaborated in Section 2.1.2, one of the main motivations for this thesis was to create a publicly available *open-source* version of the algorithm that is easily *understood* and *reproduced*, *adaptable* for other datasets and methods, as well as fast and *scalable*, mean-

ing it can be run maximally efficient on single machines but also on compute clusters, such as the IKW Grid.

This section will outline the architecture that was developed in order to achieve the aforementioned results. The resulting pipeline is the result of a lot of trial-end-error, but fulfills all of the aformentioned criteria, dealing with vastly differing sizes and kinds of datasets, minimizing runtime wherever feasible and allowing for a multitude of parameters at every step of the process.

The rest of this section will go into further detail regarding the architecture of the resulting code-base.

3.3.1. Implementation

The associated program is written by the author of this work and licensed under the *GNU General Public License* (GNU GPLv3). The source code is written in the Python Programming Language and available digitally on GitHub.[47]

The code is a proper python-package that can be installed into any Python 3.10 environment using for example python's default package manager pip:
`pip install git+https://github.com/cstenkamp/derive_conceptualspaces.git@main` .
It can then be run using `python -m derive_conceptualspace <COMMAND>`.[48] For more information on how to invoke the code base with these commands it is referred to Section A.2

To guarantee reusability of this code-base, there is also a *Dockerfile*[49] that allows to easily create a *Docker-Container*[50] from it.[51]

3.3.2. Modularity

The developed algorithm consists of clearly divisible components (as demonstrated in Figure 3.5), where the runtime for each of the steps is roughly in the same order of magnitude. All of the aforementioned (Section 3.2.1) steps are itself algorithms with many hyperparameter each. Furthermore, the framework described here does not even require particular algorithms for the individual components, but rather a classes of algorithms like *dimensionality reduction techniques*. This means that in practice, there is a combinatorical explosion of settings and hyperparameters that must be experimented with in order to find the best-performing one. Because of the clear modularity of the algorithm however, many of these become only relevant in a later step of the pipeline.

[47] Source code: `https://github.com/cstenkamp/derive_conceptualspaces/`
Source of this Document: `https://github.com/cstenkamp/MastersThesisText/`
Compiled Document: `https://nightly.link/cstenkamp/MastersThesisText/workflows/create_pdf_artifact/master/Thesis.zip`

[48] The command `python -m derive_conceptualspace -help` gives a peak into what sub-commands can be used

[49] `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/Dockerfile`

[50] `https://www.docker.com/resources/what`

[51] A Container can be thought of as a lightweight virtual operating system, in which the codebase is bundled together with all required dependencies, libraries and configurations, enabling users install this software on any system without having to download or install anything besides this container, irrespective of operating system or software versions on the host OS. For more info about the container, it is referred to `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/doc/docker_intro.md`.

Due to this, it is reasonable to make the architecture as modular as possible, storing interim results before every step, such that two parameter-combinations that differ only in e. g. the fourth step of the pipeline can share the intermediate results up to that point, keeping the required computation to a minimum.

The design principle of maximal modularity is the cornerstone of the developed pipeline. All of the interim results store the configurations that were required for the respective algorithm (and forward the ones of the input-files they transformed), as well as the created output and plots. When there are different possible algorithms for a step, it is ensured that its result are of the same format, as required by the next step. Many of the individual steps generate additional plots that can be used as sanity-checks to quickly inspect if the results so far are reasonable.

Workflow Management

A pipeline where multiple intermediate files for different parameter-combinations are created introduces the problem of *dependency resolution*: Ultimately, there is supposed to be one final file for every combination. This file however relies on intermediate files, which in turn rely on intermediate files. To resolve these dependencies, there are many existing **Workflow Management Systems**. For this thesis, **Snakemake**[52] [20] seemed the right choice.

Snakemake defines a small comprehensible domain-specific language ontop of python. With this, a workflow is described in terms of individual **rules**, each of which defining how an **output** is generated from several **inputs** using code or shell-commands. Through **wildcards**, these rules can be generalized and hyperparameters introduced [20]. The job of Snakemake is to infer a DAG from these, finding for every rule in the dependency tree for the demanded file an output that generates the required inputs, and to create jobs for all required instanciations of the wildcards if the required files are not already present. Importantly, Snakemake then also handles the inevitable scheduling problem: Due to (explicitly specified) restrictions of CPU and RAM and the nature of the unresolved dependencies, not all jobs of the workflow can be executed simultaneously. Its scheduler favors maximal utilization of CPU and parallelisation for minimal execution time [20]. Especially relevant was also that it allows to schedule these jobs on high performance clusters and computation grids, and supports among others the scheduling system SGE which is used to orchestrate jobs at the IKW grid. Configurations for the grid, like the maximal runtime or the amount of RAM and CPUs to request, can be specified per-rule as well as in special configuration files.

Snakemake was chosen because it is a lightweight system ontop of python, adding only a few lines of code to specify what inputs and outputs are created ontop of the CLI that is necessary to run and debug individual steps anyway. It is a useful tool if the workflow can be divided into rougly equally long steps which can run independently and heavily parallelized (possibly on multiple machines) with an optimal usage of resources. Its file-centric dependency resolution system allows to fill in missing steps seemlessly when working on specific configurations for later step, but on the other hand requires

[52] `https://snakemake.readthedocs.io/en/stable/`

unintuitive customization if instead configuration-files with explicit parameter-choices declare the demanded output for dynamically generated filenames. Also it unfortunately doesn't allow debugging and has a comparably small community.[53] Section A.3 shows the different ways the full pipeline can be invoked using Snakemake.

3.3.3. Modes of Execution / Use-Cases

It is possible to run the full pipeline for individual files as well as for a set of hyperparameter-configurations specified via configuration files, but also possible to run individual steps to inspect or debug the respective steps. To inspect and compare results it is possible to load all available parameter-configurations, as well as the complete history for a certain combination, listing the generated outputs and metrics. Further, individual configurations can be loaded in *Jupyter Notebooks* to generate and export plots and tables from them (like the ones used in this text). The three main ways of exectution are:

Running individual Steps per CLI is the mode of choice when working on custom steps, as it allows to attach debuggers and executes in the main thread. If a later step is executed, it is also possible to automatically generated its required dependencies using the workflow-definition. Passing configurations is possible using configuration-files, command-line-arguments or enviroment-files/-variables. For usage-examples, see Section A.2.

Loading existing Configurations for inspection especially in Notebooks, allowing to easily load a complete configuration including all its dependencies to inspect and plot (intermediate) previously created results and outputs, also allowing to iterate over several configurations to compare their results.[54] For usage-examples, see Section A.4.

Running/Scheduling a Workflow This mode is used to execute several hyperparameter-combinations at once, specified via configuration-files. Thanks to heavy integration for cluster scheduling systems, this allows for heavily parallelisation of jobs. Executing such a workflow on computation clusters is special case of this and elaborated further in the following section. For usage-examples, see Section A.3.

Running on the SGE

Due to a combinatorical explosion in the hyperparameter-space as well as the computational complexity of the algorithm, running the pipeline a sufficient amount of parameter-combinations would take several weeks on a single machine. As the IKW at the UOS owns a dedicated computation grid[55] with considerable modern hardware[56] which uses the SGE as workload manager, which is supported by snakemake, it was the obvious candidate. Snakemake encodes special configurations for clusters using *profiles*,[57] and

[53] As of 16th March 2022, there are only 1256 question tagged "snakemake" on StackOverflow (`https://stackoverflow.com/questions/tagged/snakemake`)

[54] The tables used in thesis are also automatically exported as LaTeX- code from the functions available there, as specified in their respetive references.

[55] `https://doc.ikw.uni-osnabrueck.de/content/grid-computing`

[56] Currently comprising, among many others, of 26 machines with an i7-11700 CPU and 64 GB RAM

[57] `https://snakemake.readthedocs.io/en/stable/executing/cluster.html`

there exists a profile for the Sun Grid engine.[58] Unfortunately, this default configuration does not take into account many of the pecularities of the IKW grid and it needed to be heavily customized in order to work. Foremost, all available machines to the author of this thesis have a runtime-limit of 90 minutes, which means all of the algorithm-steps that take longer than that must be able to be interrupted and gracefully shut down before getting killed and pick up the work on a new machine afterwards (including the job responsible for the workflow scheduling itself). Additionally, the arguments to request resources (such as *memory* or *parallel environments*) often differ from the documentation, and the *accounting file* which keeps track if jobs succeeded is not available to users, so a custom one must be written. Resolving these and other issues required changing the available profile heavily, so the result was open-sourced.[59].

Scheduling on such engines interestingly unveils a whole new set of "hyperparameters" that have to be optimized to use the available hardware as efficiently as possible: there are limits of how many slots are available per user, there is a fixed walltime (and interrupting and restarting leads to overhead), and the effiency of multiprocessing is not linear in the number of threads per process. Thus, depending on the size of the dataset, resources must be divided among the steps with care. The required resources of the rules are accordingly dynamically allocated in the rule-descriptions of the workflow manager.

While the code required to scalably run on the IKW-grid required much more work than expected, the result fulfills all demands perfectly, and the 64 allocated *parallel environments* (slots) are maximally utilized, while most of the complexity of the scheduling system is abstracted away.[60] The workflow is installed and run with a single (well documented) command and can be customized using explicit configuration-files. A sample output of the custom-made watcher is listed in Listing 3.1.

3.3.4. Conclusion

It was originally unexpected, but implementing an appropriate architecture for the present codebase has been a major focus of work for this thesis, and the result fulfills all of the desired design criteria:

Modularity has been the main focus in the design, so exchanging components or running individual steps is easy and intuitive.

Scalability is reached thanks to massive parallelisation wherever possible as well as a professional workflow management system that is perfectly adjusted to the available cluster engine but also highly customizable for other engines.

[58] `https://github.com/Snakemake-Profiles/sge`

[59] The resuling Snakemake-Profile is available and documented at `https://github.com/cstenkamp/Snakemake-IKW-SGE-Profile` Note that it is heavily customized to the specific engine and thus includes explicit machine names or runtimes. This repository also contains convenience-terminal-commands to inspect failed pipeline-steps or to show the current progress of the current run. A sample output of the latter is presented in Listing 3.1. Furthermore it contains `.sge`-files and shell-scripts to schedule or run a requested workflow (see Section A.3)

[60] To the best of the author's knowledge, no attempts going beyond simple `.sge`-files as job-descriptions were attempted on the IKW-grid before, and much of the available documentation turned out to be false information (as consultations with the grid's administrator have shown).

Reproducibility and Adaptability are guaranteed by stringent encapsulation of components, completely automating the full data-analysis-pipeline, open-sourcing the code as proper package and containerization of the entire codebase for guaranteed and worry-free setup on any machine or compute cluster. The exact hyperparameter-combinations of [1–3] are included (see Section B.3), allowing to re-create the original papers using this code-base. Running the code on new datasets is documented and can be done in a matter of minutes. Extending or exchanging steps of the pipeline is seamless due to a consistent and understandable data schema, and pre-existing analysis-notebooks can easily create informative plots and figures.

Transparency and Understandability are ensured due to rigorous documentation[61] (among others in this thesis) at any level of detail, from rough descriptions to concrete code-examples. Code, documentation and used data are publicly and easily available. Many analyses are included with the source-codes, for example allowing to visualise all steps of the process that can work with arbitrary numbers of dimension interactively in 3D. Code, data and configurations are clearly divided. All steps of the pipeline are very explicit about the used configurations and dependencies (making them traceable) and generate output at configurable levels of verbosity. All intermediate output can be re-accessed using helper commands.

```
Scheduler: Task-Id 736317, active for 0:03:47 on ramsauer
Active Jobs:
  Job 736319: does preprocess_descriptions_notranslate for 0:03:17 on phobos with 1 proc
      Creates siddata2022/de_debug_False/mfauhtcsldp_onlyorig_minwords80/pp_descriptions
      Progress: Lemmatizing Descriptions    :  50%|      | 10843/21643 [01:26<01:03,
  Job 736322: does create_dissim_mat for 0:05:32 on rigel with 3 procs.
      Creates siddata2022/de_debug_False/mfauhcsd2_onlyorig_minwords80/embedding_tfidf/d
      Progress: Creating dissimilarity matrix:   3%|          | 6/200 [00:47<20:55,  6.4
  Job 736323: does create_candidate_svm for 0:02:45 on beam with 6 procs.
      Creates siddata2022/de_debug_False/mfauhtcsldp_onlyorig_minwords80/embedding_tfidf
      Progress: Creating Candidate SVMs [5 procs]:   0%|          | 0/6912 [00:00<?, ?it
  Job 736324: does create_dissim_mat for 0:01:17 on cippy02 with 3 procs.
      Creates siddata2022/de_debug_False/mfauhcsd2_onlyorig_minwords80/embedding_ppmi/dis
      Progress: Creating dissimilarity matrix:   3%|          | 6/200 [00:43<20:16,  6.2
  Job 736327: does create_dissim_mat for 0:01:02 on cippy18 with 3 procs.
      Creates siddata2022/de_debug_False/mfhcsd2_onlyorig_minwords80/embedding_tfidf/diss
      Progress: Creating dissimilarity matrix:   2%|          | 3/200 [00:20<22:40,  6.9
  Job 736329: does create_dissim_mat.83.sh for 0:00:17 on vr6 with 3 procs.
  Job 736330: does create_dissim_mat.100.sh for 0:00:17 on bunda with 3 procs.
  Job 736332: does extract_candidate_terms for 0:00:02 on ocular with 1 procs.
      Creates siddata2022/de_debug_False/mfhcsd2_onlyorig_minwords80/candidate_terms_tfi
Scheduled Jobs:
  Job 736333: will do create_candidate_svm.151.sh. scheduled for 0:26:14
  Job 736334: will do create_candidate_svm.136.sh. scheduled for 0:25:15
Finished Jobs:
  Job 736318: did preprocess_descriptions_notranslate at 00:35:57 on dodo, creating sidd
  Job 736321: did preprocess_descriptions_notranslate at 00:36:12 on altair, creating si
Failed Jobs:
  Job 736320: failed at preprocess_descriptions_notranslate
  Job 736328: failed at extract_candidate_terms.43.sh
```

Listing 3.1: Sample terminal output of the custom watcher running a full configuration on the IKW-grid. The script lists the currently running jobs continously, including their progress and runtime and informs of finished or failed jobs. Another script summarises the progress in snakemake's dependency-graph.

[61] https://github.com/cstenkamp/derive_conceptualspaces/tree/main/doc

3.4. Evaluation Metrics

Two goals were stated in this thesis' introduction: To implement a reliable software-architecture that successfully replicates the works of Derrac & Schockaert [1], and figure out if their methodology also works for the domain of educational resources. Given that there is not one single correct target that the algorithm needs to optimize for, there are also no obvious measurable metrics that can be applied straight-forward to test the performance of the algorithm and no obvious *optimal results*. As the quality of the desired results is defined purely over its subjective appeal to humans[62], the only evaluation method that could quantify that would be a large-scale study with human evaluators, which unfortunately lies outside the scope of this thesis.

Instead, one has to rely on qualitative analysis of certain produced features as well as proxy metrics. This section explains what kind of results have been chosen to represent the algorithm's performance as well as why these results are suitable proxies. Furthermore it is important to test if the architecture itself works reliably, which will be tested by generating results for the placetypes-dataset and comparing them with those of the literature.

Specifically, we are interested in the following questions:

1. Is the implementation correct? / Can it replicate the results of the original implementation?
2. Is the algorithm able to cope with the Siddata-dataset despite it's different size and features?
3. Does the general methodology work for the domain of educational resources?
4. What combination of components and hyperparameters leads to the best results?

The first question is easily answered by applying the same evaluation that [1–3] used and comparing their results with ours. To answer question two, we will compare the quantity and quality of some interim results of the algorithm being applied to placetypes with their counterparts for the Siddata-dataset. Question three and four rely on some quantifiable notion of what constitutes a *good* result, so this is the first question that needs to be answered to know if the algorithm produces meaningful results.

3.4.1. Proxies hinting at meaningful results

The goal of the algorithm is to find semantic directions in the data and re-embed the entities into a vector space spun up by these interpretable human concepts as its axes. Thus, to evaluate if the emerged features are semantically meaningful, it can be measured if any known *natural categories* are among the detected semantic directions or similar to them. More specifically (as the algorithm works with BoWs which lose any information about synonymity and similarity of tokens) it is enough to answer if *terms accurately predicting* such are among the detected semantic directions. This method requires that

[62] [35, p. 133]: "It does not seem possible (nor desirable) to formally define what constitutes a good facet, a typical problem in unsupervised learning"

the dataset is annotated with the respective concept, as that will be the respective classification target. The only easily obtainable target for the Siddata-dataset is a course's faculty, the targets for the datasets of [1–3] are listed in Table 3.3.

Semantic Classifiers This evaluation method is used extensively in [1], who evaluated the practical usefulness of the created semantic directions by creating many different classifiers which only receive the entity's rank w. r. t. the detected semantic direction as input. In line with the analogy of reasoning in conceptual spaces (see 2.3.2), all their created classifiers correspond to commonsense reasoning patterns and are accordingly linked to intuitive explanations, such as a 1-nearest-neighbor classifier linked to the explanation *"Y is in the same class as X because Y is closest to Y"*. Using these, they evaluate if their derived relations are sufficiently accurate to allow for classification that is comparable to standard classification approaches while also allowing to give intuitive explanations that are compatible with high-level human reasoning such as interpolation and a fortiori inference.

Shallow Decision Trees Instead of a full suite of commonsense reasoning based classifiers, [2] and [3] train depth-limited DTs [55] on the entities' feature-based representations. DTs are supervised learning models that predict the target value by inferring binary *if-then-else* decision rules from the feature vector in a tree-like structure. The trees are constructed using the CART algorithm, which at every level finds the combination of attribute and threshold value that maximizes the prediction accuracy for the subset that falls into the respective tree's branch and partitions the attribute accordingly, to do the same on resulting subtrees (thus maximizing *information gain*). Restricting the tree's depth ensures that only the features that are most important for the decision are selected. The tree then classifies by finding the most important features for the decision, thus finding the terms that most accurately predict the demanded concept in an intuitive and interpretable way.[63] Accordingly, we check if a depth-restricted tree that can accurately predict the demanded property reasonably well can be found. If so, that would provide evidence that the semantic directions correspond to actual human concepts. The performance is measured by checking their classification on a held-out test dataset seperately for each of the target's classes in a one-vs-rest fashion. To get an estimate for the maximally possible classification performance only on basis of the text corpus, it can be compared to state-of-the-art text classification techniques such as the one described in Section B.2.2.

Unfortunately, this does not test if the names of the discovered semantic features are meaningful. It should further be noted that regardless of the classifier, only those features that predict the existing classificiation targets can be tested with this method, which in the case of the Siddata-dataset means that even if 10 features perfectly correspond to the respective faculties, all other features are still unaccounted for. In order to get additional targets, we will also compare the classification into the DDC-categories as deteced by SidBERT (see Subsection 2.4).

[63] Sample depth-1 decision trees are visualised in Figure 5.1 The decision-rule of a depth-3 decision tree corresponds to checking if a 3D-vector falls into one of $2^3/2 = 4$ boxes, as visualised in Figure C.3.

Recovering entities from salient directions Re-embedding the entities into a conceptual space involves dimensionality reduction and thus necessarily goes along with a loss of information. Apart from testing the meaningfulness of the directions, it can be tested if any relevant information was lost. To account for that, we will test if it is possible to recover the original entities using only a subset of the most salient generated features. For that, we will also analyse under which conditions separate entities fall onto the same position in the semantic space. This also provides evidence about how far the samples are distributed in the vector-space. The linear classifiers used to create semantic directions profit from data that is split far apart instead of being on a compact hypersphere which due to the *curse of dimensionality* fills only a negligible volume of the according vector-space.[64]

3.4.2. Scientific Qualitative Analysis

Regardless of the performance of proxy metrics, the most important measure of the algorithm's quality is its subjective appeal, which can only be assessed by a qualitative analysis. Unfortunately, subjectively evaluating at a subset of the resulting features and samples is, even with the best intentions, also prone to *cherry-picking*, as the selected samples may not be representative. Despite of that, a qualitative analysis provides important insight into the algorithm. To alleviate these problems, it is reasonable to use the scientific method and ask the question: *"Before looking at the data, seeing what kind of results would make us think the algorithm correctly does what we hope it does?"*

Specifically, we will try to find evidence for the following questions:

Are intuitively appealing phrases among the semantic directions? Given the task of manually embedding courses into a semantic space, there are some intuitive candidates one may think of that capture some important aspects of a course. For example, a word like *"computer"* hinting at computer-science related courses. Other obvious candidates that will be checked include *"mathe", "recht", "musik", "management", "literatur", "sprache", "psychologie", "wirtschaft", "geographie", "schule", "kultur", "wissenschaft", "sport"*.

Are top ranking courses for the directions convincing? We can not only classify courses, but also rank them according to *how much* they have a given property. To test if this is a good measure, will be to look at the courses that score highest for those dimensions that score highest for the directions that best encode one of the faculties and see if they are convincing *extreme* examples of the respective faculty or property.

Are embeddings of known similar entities close? An embedding that would put similar courses far apart from each other would not capture human intuition. Accordingly, the embeddings of exemplary courses such as *Informatik A* and *Informatik B* can be verified. In particular it can be checked if Mikolov *et al.*'s [12] findings that vector-arithmetic corresponds of embeddings matches semantic content (see Equation 1.1) by checking if *vec(Codierungstheorie und Kryptographie) - vec(mathe) + vec(informatik)* $\approx$ *vec(Kryptographische Methoden in der Informatik)*

[64] cf. `https://en.wikipedia.org/wiki/Curse_of_dimensionality#Distance_function`

4. Results

Note that sometimes the extracted dimensions or precise values of some metrics differ slightly. Mostly this is the case if a particular result comes from a task where it was optimized for a different criterion. Where that is not the case, it was due to missing random seeds that led to different train-test configurations in the final decision tree classification. This is only an artifact of the evaluation and the underlying embedding is unaffected by this, and generally demonstrates that slight variation leads to results that are different, but still of high quality. All results reported here come directly from the results as evaluated, which can also be verified in the respective repository[1]

This section summarises the results that the described algorithm achieved on the described datasets according to the described metrics. Before going into detail about the performance on the Siddata-dataset, a brief summary of the results on the placetypes-dataset serves to demonstrate if the specific implementation can achieve comparable results to [1–3], thus putting the other results into perspective in terms of what the algorithm can realistically achieve on dedicated high-quality datasets.

Due to the implemented algorithm being unsupervised, there is no explicit target value for each of the considered samples, making it impossible to straight-forwardly apply well-known ML metrics such as Accuracy or F-1 score. What this algorithm tries to achieve is a lot *fuzzier* than in the realm of classification: The end-goal of it is to embed the given entities into a vector-space that consists of semantically meaningful directions, so the only actual metric would be a comparison checking if the respective categorization here corresponds closely to human judgement. To do that, the best evaluation is likely a study that asks for feedback of users that see the results of a developed system.[2] Derrac & Schockaert [1] performed crowdsourcing experiments on CrowdFlower,[3] asking users among other tasks which of several candidates could best describe the difference between two movies. They also compared their results with those of running the supervised algorithm of [16] on a subset of their data. Furthermore they tested if the explainable classifiers generated from this algorithm (see Section 2.3.2) "help users spot incorrect classifications" [1, p. 48], as well as if their algorithmic classification corresponds to human judgement.[4] While similar studies could be done in the Siddata-DSA [8] without additional costs, carrying these out is outside the scope of this thesis.

4.1. Replicating results for the placetypes-dataset

To check if the implementation correctly produces the claimed results, it was applied to Derrac & Schockaert's [1] placetypes-dataset and its results compared to those of the literature. Figure C.1 in Appendix C shows a two-dimensional t-SNE-embedding of the

[1] `https://github.com/cstenkamp/MAAnalysisNotebooks`

[2] An example for such a system is the *Movie Tuner* interface from [16], reprinted as Figure 1.1.

[3] `http://www.crowdflower.com`

[4] The task was set only for classification into OpenCYC and Foursquare Taxonomies of the placetypes-dataset (see column '**classification classes**' in Table 3.3).

original representations of [1], colored by their GeoNames-class. This figure indicates that only few of the entities are linked with a class and that the embeddings barely cluster when compared to their embeddings for the movies-dataset (displayed in Figure C.2).

Both [2] and [3] report the performance of depth-1, depth-3 and unbounded decision trees classifying an entities' category according to the Placetypes- and GeoNames-taxonomy. Table 4.1 lists their results as well as some of their baselines in comparison with the results of this work. As shown in the table, this implementation outperforms the previous results for all classification-task-configurations. Table 4.2 shows the results of different configurations to generate the decision-tree classification. In contrast to the results reported in Table 4.1 which are optimized for their respective classification target, these results are from a single parameter-combination and robustly reproducible.

		Literature baselines			**Literature results**			
Target	**Cls**	**Ran**	**LDA**	**BL**	**[1]**	**[2]**	**[3]**	**this work**
GeoNames	**D1**	0.23	**0.34**	-	**0.32***	0.32	0.28	**0.51**
	D3	0.27	0.32	-	0.31*	0.31	**0.34**	**0.54**
	DN	-	0.27	0.2	**0.37**	0.24	-	**0.46**
	Any	-	-	0.36	**0.41**	-	-	-
Foursquare	**D1**	0.39	**0.55**	-	0.38*	0.41	**0.45**	**0.50**
	D3	0.5	0.48	-	0.42*	0.44	**0.57**	**0.58**
	DN	-	0.47	**0.53**	**0.53**	0.42	-	**0.57**
	Any	-	-	0.72	**0.73**	-	-	-

Table 4.1.: F1-scores of classifiers predicting GeoNames- and Foursquare-labels for three baselines, [1–3] and this work. **Cls** column encodes the classifier: **D1/3** are DTs of depth 1/3, **DN** unbounded DT. Condition **Any** refers to the best of [1]'s semantic classifiers. Baseline-columns: **Ran** is the *Random* baseline as reported by [3], **LDA** is LDA as reported by [2], **BL** is the best baseline-condition from [1]. Columns **[1], [2]**, **[3]** encode the best reported scores per publication. Starred values in column **[1]** refer to results that [2] reported for the configuration of [1] for conditions not covered by the latter. Final column reports the results of this work (with balanced samples and 1vsRest-condition; unlike Table 4.2, this reports the respectively optimal parameter-configuration just like the best reported configuration from [1–3]) with a literature-consistent train-test-split of 70-30. A longer version of this table listing more configurations per publication can be found in the Appendix as Table C.2.

1vsRest	Depth Balanced	Any	1	2	3
False	**False**	**0.419**	**0.494**	0.471	0.496
	True	0.394	0.109	0.298	0.342
True	**False**	0.413	0.314	0.331	0.378
	True	0.382	0.487	0.505	0.506
T,U	**False**	0.218	0.103	0.116	0.152
	True	0.195	0.284	**0.294**	**0.292**

(a) GeoNames

Any	1	2	3
0.527	**0.358**	**0.453**	**0.540**
0.535	0.128	0.189	0.281
0.513	0.229	0.453	0.494
0.548	0.488	0.563	0.558
0.218	0.103	0.116	0.152
0.195	0.284	0.294	0.292

(b) Foursquare

Table 4.2.: F1-scores of various decision trees predicting GeoNames- and Foursquare-labels. All scores result from 5-fold cross-validation averaged across 10 random seeds, all from the parameter-configuration. Rows correspond to different hyperparameters for the DT: If not *1vsRest*, a single tree must predict all classes at once (max. 2^{depth}), otherwise performances of one classifier per class are averaged. Middle rows report scores weighted by class frequency, bottom rows (Condition **T,U**) unweighted averages of the same classifiers. If *Balanced*, class weights are inversely proportional to class frequencies. Results highlighted in green are respectively optimal scores, bold results are optimal scores if class-score-weighting is forbidden.

4.2. Dataset differences

Having established that the implementation works correctly for the domain of placetypes, we will now check if the quantity of some key characteristics produced as interim results of the algorithm differ across domains. As discussed in Section 3.1, our dataset differs in some properties from those of the literature. Because interim resuls may already indicate the performance of the algorithm, looking at them provides useful information that can be considered in the performance evaluaution. Table 4.3 contrasts the number of feature vectors, candidate terms and cluster elements for the movies-, placetypes- and Siddata-dataset.

Note that the first two rows display sample results of Derrac & Schockaert's [1] original implementation for the domains of movies and placetypes as it was uploaded by the authors. Looking at the number of candidates and the number of those with a kappa-score of at least 0.1, we see that in the case 200D × placetypes, 21 819 out of 21 833 candidate terms had a kappa-score of at least 0.1 and were thus considered *semantic directions*.

The Siddata-dataset consists of more but shorter texts associated with an entity (visualised in Table 3.1), which is why there are much less words in the respective texts that exceed a given df for the Siddata when compared to the originally used ones (visualised in Table 3.2). The algorithm uses a SVM to split entities that contain a phrase from those that do not, which performs bad for heavily imbalanced class sizes. Figure 4.1 shows the distribution of texts per candidate as a histogram. It shows that 90% of the 10 060 candidates occur in 26 to 375 of the 11 601 documents. The median number of

			Kappa-Scores						Cluster sizes			
			$\kappa \geq 0.1$			$\kappa \geq 0.5$			10^{th}		90^{th}	
	Terms	**Cands**	**50D**	**200D**	**Sum**	**50D**	**200D**	**Sum**	**50D**	**200D**	**50D**	**200D**
movies [1]	589 727	22 903	9 429	13 916	13 918	-	-	444	46	10	467	61
placetypes [1]	746 180	21 833	20 246	21 819	21 832	-	-	697	30	9	1 616	115
Siddata	163 285	10 060	3 010	5 016	5 937	334	1 008	481	7	2	30	16

Table 4.3.: Distributions of some interim algorithm results for the considered datasets. The first two rows are extracted from the results uploaded by Derrac & Schockaert [1], the last from this implementation. First column are all unique *Terms*, second column those considered **Candidates** by the algorithm. Columns under **Kappa-Scores** indicate the number of candidates that exceeded the respective threshold (where reported). **Sum** column is the size of the set union of the candidates exceeding the respective threshold over all dimensionalities, indicating robustness (optimally $\max(400, 200, 100, 40) = 400$, worst-case $400 + 200 + 100 + 40 = 740$). **Cluster sizes** refers the 10^{a}nd 90^{p}ercentile of the cardinality of all respectively extracted clusters.

documents is 49, meaning that for half of the keyphrases only 0.42% of the samples are in the positive class.

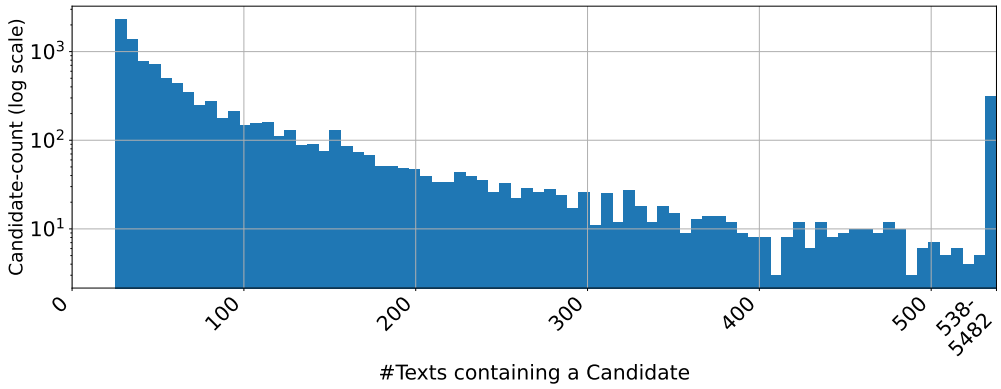

Figure 4.1.: Distribution of texts per candidate (log scale), cut off at the 97^{th} percentile. Df-threshold for a candidate is set to 25, yielding 10 060 candidates for 11 601 documents. Median number of documents per candidate is 49 and for the 95^{th} percentile 375. 2595 candidates occur in at least 100 descriptions.

These results show that the algorithm produces much less candidates for our dataset compared with the originally considered ones. Because of that, the subsequent steps of the algorithm have a less rich representation to choose from, decreasing the chance to achieve good performances downstream.

4.3. Results for the Siddata-dataset

As previously described, the primary method used here to check if the described methodology works for the domain of educational resources is to check if low-depth decision trees trained on the extracted semantic directions of the Siddata-dataset can classify a courses' faculty. Before doing that however, it is important to first validate if it can reas-

onably assumed that the faculty *can generally* be extracted from only the descriptions associated with the entities.

Extracing faculties without the algorithm

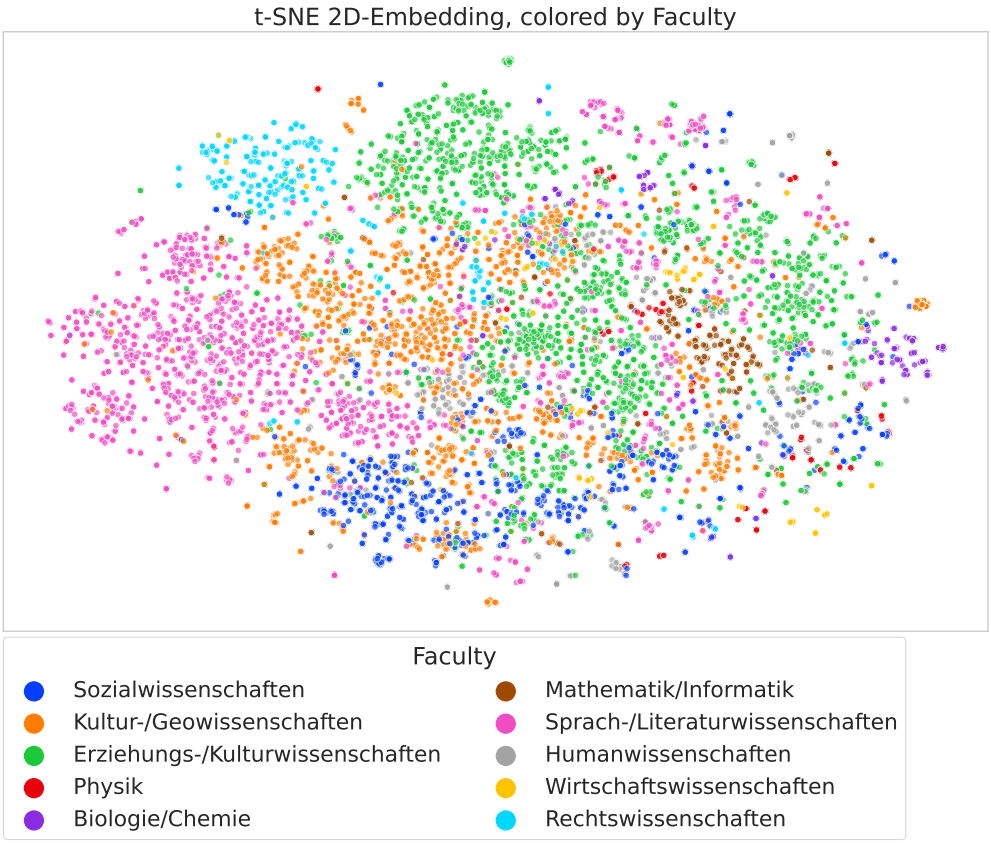

Figure 4.2.: 2D Visualization of the Course-dissimilarity matrix, generated with t-SNE. See `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/notebooks/text_referenced_plots/visualise_embeddings.ipynb` for the origin of this plot as well as a 3D-plot on unaltered 3D-MDS-data that does not rely on t-SNE.

To see if it is possible to extract any kind of structured data from the unstructured course descriptions, a BERT-based Neural Network classifier was trained on the dataset, classifying the subset of courses that are for the UOS to the faculty they belong to. The architecture of the classifier is described in Appendix B.2.2.

The classifier trained for 12 epochs before *stopping early* due to its performance on the test-set decreasing. It achieved an accuracy of **85.19%** for the test-set (94.13% on the training set), providing strong indication that the faculty can be considered a latent property of its description. Because BERT is one of the best general language classifiers to date [15], this accuracy will be considered the upper boundary for the results of our algorithm.

Figure 4.2 displays a t-SNE-representation of the dissimilarity matrix generated from the normalized angular distances (Equation 3.1) of the BoW-representations generated from

the courses (only those from the UOS) A first comparison of this scatter-plot and its placetypes-counterpart (Figure C.1) reveals that Siddata appears to have more homogenous clusters.

From this distance matrix, the algorithm subsequently creates an embedding using the MDS-algorithm to afterwards train multiple SVMs for each of the extracted candidate-dimensions before re-embedding the entities into a new space where the dimensions encode the rank for each of the extracted features. To evaluate the algorithm, we will check if it is to be able to classify the faculty with a decision tree that uses between one and maximally $2^3 = 8$ of these features. To provide a context for the resulting performance, we will first look at a classification which does not use the feature-based representation, but a raw low-dimensional embedding without class-specific dimensions. This can be seen as the lower boundary of what a classification without selecting the most relevant dimensions but instead use an optimal but general three-dimensional space can achieve. A classification-performance of low-level decision trees that use only the most important features from all available ones that is higher than this thus provides evidence that the detected features encode important distinctive properties. Figure 4.3 visually represents the result of the classification using a linear SVM on a three-dimensional space generated as result of MDS on the dissimilarity matrix of the entities. Averaged over all Faculties, this classifier reaches a weighted accuracy of 64.3% (unweighted accuracy 69.0%, weighted F1: 0.414, unweighted F1: 0.269). Note that these values are reached by training on the full data without a separate testing-set.

Extracing faculties with DTs based on the algorithm

Having established upper and lower boundary for what can reasonably be expected from the algorithm, let us finally look at the performance of its decision-trees, to gain evidence if human concepts are encoded in its extracted features. Table 4.5 lists average accuracies per faculty for decision trees created on the basis of a single hyperparameter-configuration that was selected to achieve high performance on average. The respective values are the average and standard-deviation of runs with 5-fold crossvalidation each for trees of various depths. The results show high accuracies across all faculties even for low-depth trees, but the faculty *Humanwissenschaften* displays a high standard deviation for that condition.

Depth	**1**	**2**	**3**	**any**
Accuracy	0.056 ± 0.008	0.078 ± 0.042	0.196 ± 0.017	0.584 ± 0.015
F1	0.072 ± 0.006	0.125 ± 0.014	0.199 ± 0.007	0.513 ± 0.020

Table 4.4.: Robust scores of a well-performing configuration for classifying all faculties at once. The reported results are mean and standard deviation from the result of ten runs with 5-fold crossvalidation each.

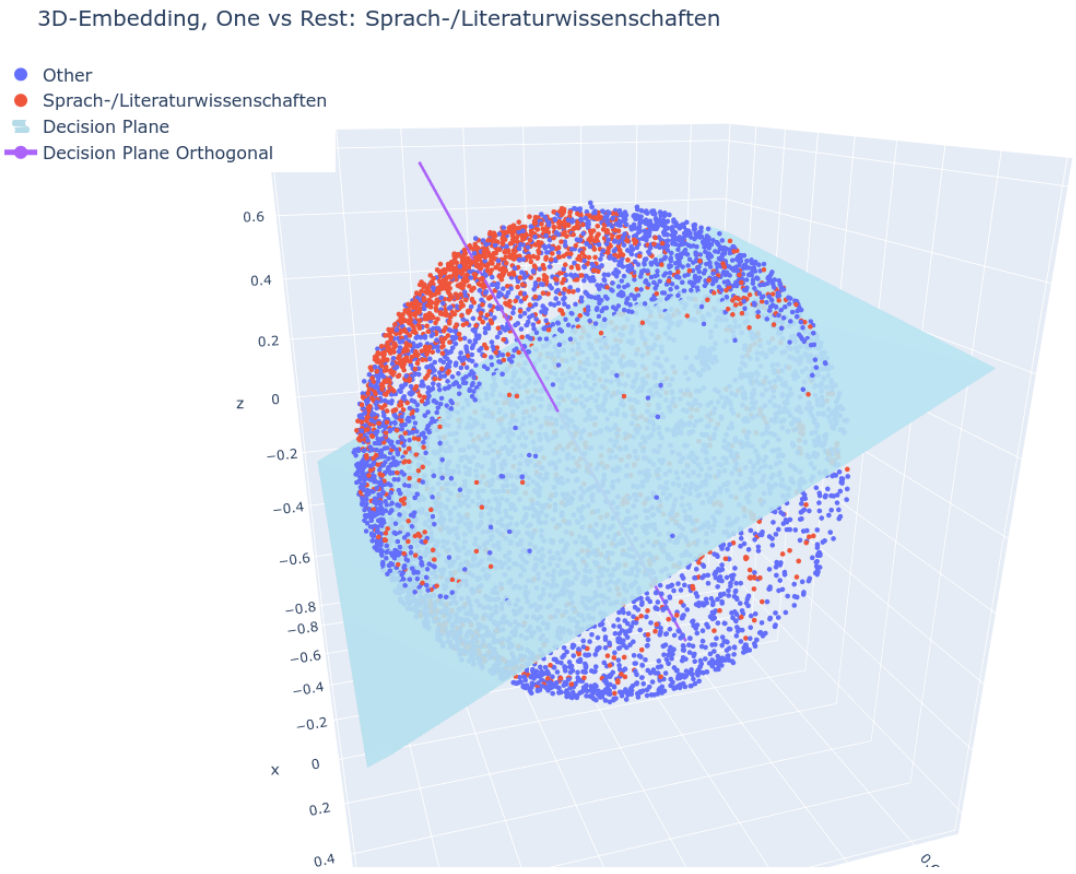

Figure 4.3.: A possible Hyperplane on a 3-Dimensional Embedding. The SVM depicted here reaches an Accuracy of 67.9% (Precision: 39.8%, Recall: 70.5%). visualise interactively: `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/notebooks/text_referenced_plots/visualise_embeddings_mds.ipynb`

Depth	1	2	3	unbound
Sozialwissenschaften	0.831 ± 0.021	0.811 ± 0.033	0.780 ± 0.028	0.918 ± 0.006
Kultur-/Geowissenschaften	0.806 ± 0.010	0.728 ± 0.066	0.813 ± 0.022	0.873 ± 0.008
Erziehungs-/Kulturwissenschaften	0.791 ± 0.010	0.824 ± 0.010	0.828 ± 0.020	0.868 ± 0.008
Physik	0.747 ± 0.054	0.770 ± 0.033	0.818 ± 0.038	0.983 ± 0.003
Biologie/Chemie	0.787 ± 0.044	0.838 ± 0.049	0.890 ± 0.036	0.982 ± 0.004
Mathematik/Informatik	0.866 ± 0.031	0.844 ± 0.051	0.882 ± 0.038	0.978 ± 0.004
Sprach-/Literaturwissenschaften	0.832 ± 0.009	0.832 ± 0.009	0.862 ± 0.009	0.902 ± 0.008
Humanwissenschaften	0.630 ± 0.130	0.768 ± 0.103	0.781 ± 0.093	0.949 ± 0.005
Wirtschaftswissenschaften	0.903 ± 0.015	0.910 ± 0.034	0.924 ± 0.018	0.989 ± 0.003
Rechtswissenschaften	0.948 ± 0.031	0.894 ± 0.015	0.952 ± 0.011	0.985 ± 0.003
Mean (weighted)	0.814 ± 0.035	0.822 ± 0.040	0.853 ± 0.031	0.943 ± 0.005
Mean (unweighted)	0.810 ± 0.020	0.806 ± 0.031	0.835 ± 0.023	0.901 ± 0.007

Table 4.5.: Robust accuracies per faculty of a well-performing configuration. The reported results are mean and standard deviation from the result of ten runs with 5-fold crossvalidation each.

Keep in mind that accuracies often make the situation look better than it is. The weighted average F1-scores are between 0.505 (depth 1) and 0.710 (unbounded). For the full table of F1-scores it is referred to the appendix, specifically Table C.4.

Compared with the results of the BERT-based classifier (85.19%), the achieved results are suprisingly competitive, with the weighted mean of trees of depth one and two only slighly below that, and deeper trees achieving even higher accuracies. Importantly however, the results reported here are those of a sepearate classifier per faculty (*1vsRest*), differentiating only between the respective faculty as positive class and all other faculties as negative class. As will be further elaborated in the discussion, this is the only realistic way of doing it: A binary tree of depth one can differentiate between maximally two classes and thus maximally achieve a classification by perfectly classifying the most frequent class and labelling all samples as the second most frequent class, which in the case of faculties is $(2011 + 1665)/7081 = 51.9\%$. Results for such trees are reported in Table 4.4.

Recovering entities from salient directions

After having tested if our semantic embedding captures at least one prominent latent topic of the data, we will additionally check if the produced embedding adequately captures enough of the general variability of the data. For that, we check if it is possible to recover entities from the salient directions, to ensure that no important relevant information was lost when mapping them to their semantic embedding. To get a general understanding, let us consider under what circumstances multiple different entities will fall towards the exact same coordinates.

To get an estimate of the importance of each dimension of the embedding, we started with a full matrix where each entity is defined via each semantic direction. Starting with a 200-dimensional space, we removed random directions. Further we applied different levels of discretisation to the remaining dimensions. We wondered what amounts of discretisation and removal of dimensions is necessary for multiple entities to fall onto the same category. Each combination was done multiple times, each time removing other random directions, and the percentage of duplicates in the data was counted. Table 4.6 displays the result of this.

	2	**11**	**23**	**116**	**232**	**1160**
#dims	(5800)	(100)	(50)	(10)	(5)	(2)
3	100%	99.85%	68.98%	6.85%	5.25%	3.97%
5	100%	24.91%	9.79%	5.25%	4.72%	3.42%
10	100%	10.18%	6.90%	4.67%	4.23%	2.11%
20	100%	7.41%	5.58%	4.23%	3.52%	0.79%
50	99.97%	5.46%	4.72%	3.19%	1.89%	0.05%
100	99.88%	4.80%	4.18%	1.94%	0.65%	0%
200	99.55%	4.36%	3.44%	0.67%	0.09%	0%

Table 4.6.: Duplicates per combination of dimensionality and amount of categories per left-over category. Row = number of left-over dimensions, cols = discretisation (number in parantheses is the divisor of the orginal value)

4.3.1. Qualitative Analysis

The main goal of the algorithm is not to optimally classify the respective faculties, but to find semantic directions. Thus, once it has been established that the performance is reasonably well, looking at the names of these directions is just as important. Figure 4.4 visually represents decision-trees for each of the faculties of depth one, i. e. trees that can only use a single semantic direction for their decision. With the exception of the faculties *Physik* and *Biologie/Chemie*, the semantic directions that best predict each faculty appear very relevant to the respective topic.

For trees deeper than one level, the selected features from the second level on are not necessarily the most important ones.[5] Instead it is possible to extract the most important features for a classification by looking at the respective information gain achieved by splitting it. Table 4.7 displays the resulting three most important directions for each of the respective faculties. While the details of this result will be eloborated upon in the discussion, most of the detected features appear convincing. Regarding an actual classification using these features, Figure C.3 in the Appendix displays the result of a sample classification of a decision tree with three features.

Faculty	Top 3 Directions			Accuracy Top 3	Accuracy Top 1
Erziehungs-/Kulturw.	erziehungswissenschaft	okumenisch	english for	78.02%	75.50%
Rechtswissenschaften	juristisch	bgb	bgb	95.86%	91.10%
Wirtschaftsw.	center betriebswirtschaftlich kompetenz	religionsunterrichts	design	89.65%	79.10%
Kultur-/Geow.	tourismus	gi	stadtgeographie	75.50%	77.44%
Mathem./Informatik	programmiersprache	menge	hoffnung	93.00%	91.85%
Sprach-/Literaturw.	deutsch literaturwissenschaft	sprache	okumenisch	86.71%	85.10%
Humanwissenschaften	psychologie	metaphysik	internationalisierung	86.84%	85.51%
Physik	neu entwicklung	mitarbeiterinnen	regelmassig aktiv teilnahme	78.27%	77.36%
Biologie/Chemie	aktivierung studierend	brd	berucksichtigung finden	85.22%	62.13%
Sozialwissenschaften	arbeitsmarkt	regieren	multiple	78.60%	67.63%

Table 4.7.: Top 3 directions to detect the respective faculty from the data. Note that it may also be the case that low values for the respective feature encode class membership.

4.4. Optimal Parameters

Having established the algorithm's ability to be transferred to the domain of educational resources, we will conclude this chapter with the results of our hyperparameter search. The algorithm was run with hundreds of different hyperparameter-combinations for both the placetypes- and Siddata-dataset throughout the process of its implementation and testing. As the precise hyperparameter-combination was not very imporant for our research questions, only some exemplary results for the Siddata-dataset will be reported here for the sake of brevity.

[5] A DT has separate conditions for every subtree, which means on level two there are two different features that depend on the result of the first split.

Cluster Center	Cluster Elements
opnv	befragungen, bewohner, urbanen, 00 uhr
bertolt brecht	brecht, exil, bertolt, weiss, weimarer republik, auschwitz, weimarer, ...
religionswis-senschaft	religionsunterrichts, spirituelle, religions, teil veranstaltung, religion, studienbeginn, auml ischen, ...
erkrankungen	klinischen, gesundheitlichen, krankheit, rehabilitation, pravention, aufrechterhaltung, exemplarische inhalte, ...
lineare	integrierte, berechnung, variablen, skript, ubungsaufgaben, losung, vorliegen
parteien	verfassung, demokratischen, zivilgesellschaft, semester sowie, einander, folgendem link, sozialwiss uni osnabrueck, ...
kleidung	textilien, mode, textile, parallelen
kreuz	heilige, christentum, 1957, fuhrungen
offentliches	offentlichkeitsarbeit, verwaltungsrecht, offentlichen recht, offentliches recht, europarecht, staatsrecht, teilnahmevoraussetzungen veranstaltung, ...
masterstudium	fortgeschrittene, schlusselkompetenzen, betriebswirtschaftslehre, untereinander
unterrichtspraxis	fachlichen, unterrichtsplanung, schulalltag, lehrerinnen, schulische, daz, bildung nachhaltige, ...
aristoteles	asthetik, semantik, dialoge, menschheit, sucht, benutzt, philosophische, ...
mythologie	athen, mythos, mann, griechen, ubersetzung, figur, griechischen, ...
flexibilitat	weiterbildung, schlusselqualifikation, erfolgreiches, fordert, vorlagen, strukturierte, berufsleben, ...
aktive beteiligung	teilnahme ersten, wochentliche, zweit, festlegung, ubergreifende, auml ndige, ndige, ...
unsicherheit	covid 19 pandemie, korperlich, wendet studierende, webinar, risiko, einbringen, fachlich, ...
wirtschaftlicher	industrialisierung, wachstum, republik, wirtschaftspolitik, konkurrenz, positive, russland, ...

Table 4.8.: Exemplary clusters found in the Siddata-dataset.

The reports produced in the previous section are the best ones from a final set of 165 different parameter-combinations, run over the course of three days on the IKW-grid.

Generally, we selected a parameter-configuration that reached a high accuracy on average for the aforementioned classifiation based on shallow decision trees as representatively *good* configuration. Here we compare all of the parameter-combinations that were considered - however the printed tables are shortened for a clearer overview.

As described in Section 3.2.1, a good first approximation for the quality of an embedding is to check how many candidate-terms get a kappa-score which is above the threshold of 0.5. Unfortunately, [1] and its follow-ups [2, 3] did not explictly and unambiguously state how the kappa-score was calculated. Not only does its implementation have many arguments, but also the application may vary. In general, it compares the *prototypicality* according to the classificiation decision with the quantificiation of how often the according word occurs in the entities' associated texts. The precise mechanism of has however many more parameters: for example, one can only consider those entities that have a positive quantificiation-score, such that not almost all of the candidates have a score of

zero[6]. Another open question is when considering the raw count as quantification[7], if the rankings induced by the orthogonal of the hyperplane are to be compared with the raw count, or the *ranking induced* by the count. To clarify this ambiguity, we tried out many different interpretations for this score.

Table C.5 (moved to the Appendix) shows the results of many runs with different parameter-combinations with the purpose of figuring out which combination of parameters and kappa-metrics lead to enough candidate-terms. The meaning of the different kappa-scores encoded in the columns is given in the implementation details in Table B.1. . The table shows drastically differing values for the individual scoring methods. To understand if the difference in scoring has an effect on *which* candidates are extracted, let us look at the overlap of different measures in Table 4.9.

	accuracy (10060)	**precision** (1668)	**recall** (10060)	**f1** (3155)	**r2r-d** (0)	**r2r-min** (2)	**b2b** (3052)	**dig** (0)	**c2r+** (4)	**r2r+d** (0)	**r2r+min** (239)	**r2r+max** (103)	**dig+2** (1010)
accuracy	-	0.166	**1.000**	0.314	0.000	0.000	0.303	0.000	0.000	0.000	0.024	0.010	0.100
precision	**1.000**	-	**1.000**	**1.000**	0.000	0.001	0.998	0.000	0.002	0.000	0.084	0.051	0.126
recall	**1.000**	0.166	-	0.314	0.000	0.000	0.303	0.000	0.000	0.000	0.024	0.010	0.100
f1	**1.000**	0.529	**1.000**	-	0.000	0.001	0.967	0.000	0.001	0.000	0.067	0.032	0.140
r2r-d	0.000	0.000	0.000	0.000	-	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
r2r-min	**1.000**	**1.000**	**1.000**	**1.000**	0.000	-	**1.000**	0.000	0.000	0.000	0.000	0.000	0.000
b2b	**1.000**	0.546	**1.000**	**1.000**	0.000	0.001	-	0.000	0.001	0.000	0.068	0.033	0.141
dig	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-	0.000	0.000	0.000	0.000	0.000
c2r+	**1.000**	**1.000**	**1.000**	**1.000**	0.000	0.000	**1.000**	0.000	-	0.000	**1.000**	**1.000**	0.250
r2r+d	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-	0.000	0.000	0.000
r2r+min	**1.000**	0.586	**1.000**	0.887	0.000	0.000	0.874	0.000	0.017	0.000	-	0.406	0.452
r2r+max	**1.000**	0.825	**1.000**	0.981	0.000	0.000	0.981	0.000	0.039	0.000	0.942	-	0.437
dig+2	**1.000**	0.208	**1.000**	0.438	0.000	0.000	0.426	0.000	0.001	0.000	0.107	0.045	-

Table 4.9.: Overlap of different scoring methods in percent. Numbers in parantheses list the total number of extracted samples for each scoring method. Upper right triangle encodes the cardinality of the union divided by the cardinality of column-method, lower left triangle divides by row.

This table additionally reports the raw *accuracy, precision, recall* and *f1*-scores of the classifier. The overlaps of these scores will serve to test our hypothesis that due to the different nature of the Siddata-dataset, comparing the *ranking* of the candidates leads to worse performances than comparing raw classification scores (also suggested by [2]).

After these preliminary analyses with the number of extracted candidates, let us also quickly look at differences in perforamances of the decision-trees depending on parameter-combinations. Table 4.10 lists the accuracies of DTs classifying a course's faculty per parameter-setup for some of the configurations of the final run. Specifically, this compares accuracies for several *1vsRest* decision trees of depth one with a test-set size of 33%.

[6]For the movies-dataset, a df threshold of 100 for a candidate means that up to 14 900 entities (99.33%) do not contain the entities, meaning all scores and accordingly also their rank will be zero. In constrast to that, the ranking induced by mapping of the high-dimensional and real-valued embeddings onto the separatrix orthogonal will almost always lead to unique ranks.

[7]The exact wording of Derrac & Schockaert [1] is: *"[...] measure the correlation between the ranking induced by* $\boldsymbol{v}\_\boldsymbol{t}$ *and the number of times t appears in the documents associated with each entity [...]"*. This seems to imply that they are using the raw count, even though [2, 3] write in their algorithm summary that they used the PPMI-score

			Dimensions	3		50		200	
			$Lambda_2$	0.1	0.2	0.1	0.2	0.1	0.2
Preprocessing	**Quantification**	**DCM quant**	**Metric**						
• **Sentence-wise merge** • **add titles** • **add subtitles** • **rm HTML-tags** • **lower-case** • **rm stopwords** • **rm diacritics** • **use SK-Learn**	**ppmi**	**count**	c2r+	53.24%	47.22%	48.36%	41.91%	40.87%	36.57%
			dig+2	49.66%	58.22%	74.73%	71.75%	80.32%	76.81%
			r2r+min	67.79%	65.09%	74.99%	72.61%	74.51%	73.72%
		ppmi	dig+2	63.49%	64.76%	55.78%	61.40%	55.10%	65.32%
			r2r+min	62.43%	67.83%	75.30%	75.99%	76.97%	73.84%
		tfidf	dig+2	55.81%	62.23%	75.22%	73.88%	76.34%	79.14%
			r2r+min	63.23%	62.35%	70.58%	72.60%	77.69%	72.75%
	tfidf	**count**	c2r+	44.99%	59.00%	60.27%	37.56%	56.05%	43.24%
			dig+2	45.45%	49.86%	75.75%	77.77%	76.25%	80.86%
			r2r+min	61.91%	63.94%	72.69%	76.24%	-	79.49%
		ppmi	dig+2	59.96%	61.40%	75.72%	79.54%	73.05%	74.02%
			r2r+min	53.73%	55.10%	78.20%	79.53%	80.77%	79.47%
		tfidf	dig+2	56.14%	54.78%	75.32%	76.83%	-	79.44%
			r2r+min	60.44%	59.72%	78.26%	77.56%	78.81%	77.69%
• **Sentence-wise merge** • **add titles** • **add subtitles** • **rm HTML-tags** • **Sentence-tokenisation** • **lower-case** • **rm stopwords** • **Lemmatize** • **rm diacritics** • **rm punctuation**	**ppmi**	**count**	c2r+	-	47.07%	40.11%	49.73%	44.68%	40.19%
			dig+2	58.27%	58.48%	75.51%	73.56%	78.86%	78.04%
			r2r+min	56.83%	69.00%	72.68%	72.59%	71.81%	72.63%
		ppmi	dig+2	58.72%	59.84%	69.96%	72.66%	71.10%	65.55%
			r2r+min	59.29%	67.17%	76.10%	79.40%	77.18%	80.27%
		tfidf	dig+2	62.89%	63.42%	76.44%	73.92%	75.15%	78.93%
			r2r+min	57.79%	59.07%	76.81%	76.61%	72.92%	74.68%
	tfidf	**count**	c2r+	61.90%	61.22%	46.97%	51.90%	48.99%	53.48%
			dig+2	50.29%	50.43%	79.51%	78.89%	80.32%	78.37%
			r2r+min	61.62%	57.49%	75.26%	79.52%	79.93%	78.67%
		ppmi	dig+2	60.45%	58.77%	79.95%	80.03%	79.68%	80.63%
			r2r+min	63.10%	63.56%	78.56%	79.89%	80.49%	80.70%
		tfidf	dig+2	62.14%	63.87%	77.97%	80.25%	76.66%	78.65%
			r2r+min	63.09%	62.09%	78.27%	76.41%	79.06%	80.92%

Table 4.10.: Decision tree accuracies for different parameter-combinations. (tree of depth 1, balanced, 1vsRest, 33% testset)

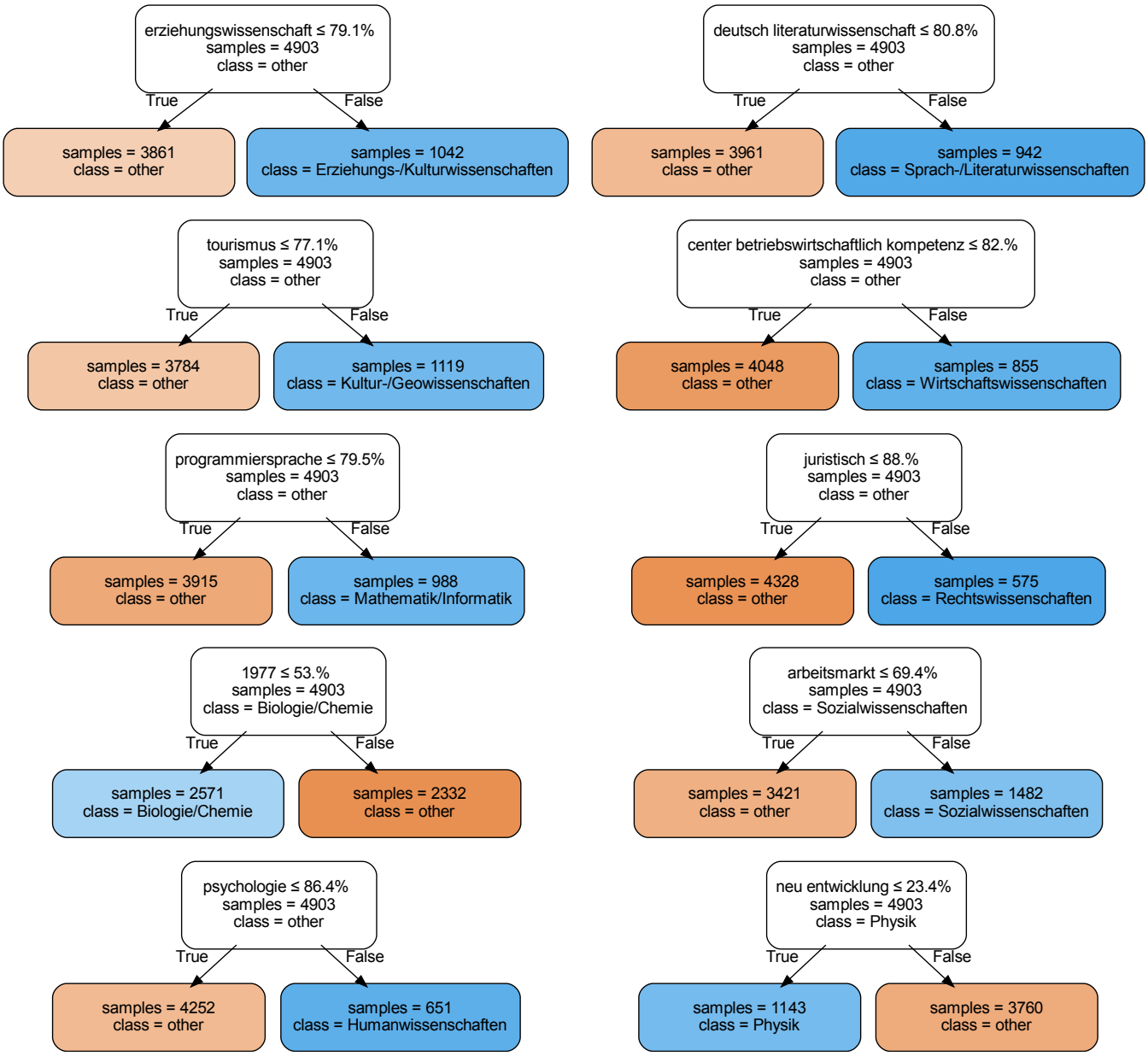

Figure 4.4.: Resulting DTs with only a single decision for each of the faculties. The white boxes show a semantic direction and the maximal rank w. r. t. to this direction for an entity to fall under the class designated by the respective left branch.

5. Discussion and Conclusion

The results generated thus far should finally answer our research questions and check if the general thesis goals were achieved. In this section, we will first interpret the generated results iteratively to be able to assess the performance of the algorithm on the dataset. On the basis of that, an answer to the general question if the methodology is applicable for the domain will be formulated. This is followed by a discussion of the algorithm in itself from the perspective of somebody that worked with it extensively. Finally the architechture is quickly discussed and the thesis concluded.

5.1. Interpretation and Discussion of results

First we evaluate our implementation by comparing its performance for the placetypes-dataset and discussing possible reasons for any discrepancies. Following that, we will discuss the results for the Siddata-dataset when compared to those of the literature to see if the algorithm can cope with the domain transfer. On the basis of which we will try to find an answer to the question if the general methodology is able to cope with the domain in question. Afterwards we will discuss the results of the hyperparameter-search and its implications on our dataset.

5.1.1. Results for Placetypes

When doing classification with decision-trees it is a design-decision to either train one decision-tree per class that just classifies if a sample is that of that class or not (*1vsRest*), or alternatively generate a single tree that must predict the exact class membership for all classes at once (*AllAtOnce*). In Table 4.2, we reported the results for both of these conditions.

The results indicate that performances for depth-limited trees are not consistently worse than unbounded trees. This is not suprising, considering that decision trees are known to be prone to overfitting, which can only really happen for unbounded trees. Similar results were reported in the work of Ager *et al.* [2].

The best results were achieved for the condiditions *balanced 1vsRest* and *unbalanced AllAtOnce*. In general the results indicate that especially for depth-limited trees it holds that for single *AllAtOnce* classifiers, balancing is bad for performance an vice versa. Considering that depth-limited ones can only predict very few classes explains why the *AllAtOnce* condition performs badly overall. For the same reason, however, these classifier benefit from unbalanced datasets - without balanced sample-weighting, these trees can just detect the most common class labels and assign them, which was shown to be the case here as well. Generally however, especially for depth-limited trees, *1vsRest* improves performance.

Explanations for good results

As shown in Table 4.1, the achieved results outperform those of the literature for the placetypes-dataset in all cases, often with a significant margin. Considering that this implementation replicates [1] without major algorithmic improvements and does not contain some of the improvements of [2, 3], this is initially surprising, so here we will discuss some possible explanations for that.

Errors Naturally, the first thing to to in this situation is to check for errors in the implementation. In following that route, however, it is important to keep in mind that errors in the actual algorithm are an unlikely candidate for an erroneously high performance. As elaborated before, the performance of the decision-tree is only a surrogate metric to evaluate the resulting semantic directions. Among others this implies that the classification target for the task is disregarded in all algorithm steps except the final evaluation with the decision trees. As long as that is given, the only realistic source of error that leads to higher-than-expected accuracies reliably is thus in this step. This does not mean that there are certainly no errors in the implementation of the rest, but as long as the classification target is not used, all these errors would only coincidentally lead to better classification results. In contrast to that, there are many sources of errors in the decision tree classification that will likely lead to unrealistically high performances such as mixing up the training- and testing set. In any case, both the algorithm itself and the decision-tree classification was triple-checked for errors and many sanity-checks were performed that all lead to the same conclusion, so from now on we will assume that the results are correct and discuss possible reasons for that[1]

One-Vs-Rest-Classification [2, 3] are both unclear if they did the former or the latter. Generally, All-At-Once would be the harder task and comparing accuracies of 1vsRest to AllAtOnce an unfair comparison. However, there are a few things that can be assumed:

Both of them report to have used the sklearn-implementation of decision-trees, just like this work. This specific implementation reportedly uses the CART algorithm [55], which only allows binary trees, where every node has exactly two children[2]. Consequently, a decision tree of depth one can only classify $2^1 = 2$ classes, whereas a tree of depth two can classify up to $2^2 = 4$ classes. Due to that, the best achievable accuracy of a perfect depth-1-tree is $\frac{||\text{samples in two most common classes}||}{||\text{samples in all classes}||}$, which is $\frac{176+74}{403} = 0.62$ in the case of GeoNames and $\frac{88+82}{391} = 0.43$ for Foursquare.[3] The latter value is lower than what [3] report, indicating that is not how the authors generated results. This would be even a lot more pronounced when classifying the movie genre, which has 100 classes.

Also semantically it is reasonable to assume to do 1vsRest: they state extensively that they are looking for a direction for *scarieness* in movies, where the genre corresponding to that *(Horror)* is only one of the genres. This kind of mapping Genre-FeatureDirectionPredictingGenre can only be found with separate trees per genre - and

[1] The code is open-source and available at `github.com/cstenkamp/derive_conceptualspaces`, and the author of this thesis is thankful for any issues.

[2] `https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart`

[3] Note that these values only hold on average, as the samples are arbitrarily assigned to the train- and test-set.

thus generally per class. Both of these reasons lead us to the assumption that they likely also did a separate tree for each of the features.

Dataset size Only a small subset of samples even *have* a class assignment (403 of 1383 in 7 classes in the case of GeoNames (see Figure C.1), 391 in 9 classes for Foursquare), and the classes are heavily imbalanced (GeoNames between 176 and 14 samples per class, Foursquare between 88 and 6). We used the classes as uploaded by [1][4], of course there is the chance that they did not make all of their data publicly available, whereas [2, 3] had access. Furthermore, that is generally a tiny dataset so the statistical power of any result here is really low - maybe it is just coincidendence

Not having some improvements of [2] The contributions from [2] were primarily the Fine-Tuning and the condition where averaged word-embeddings are used instead of MDS (denoted **AWV**). As can be seen in Table C.2, these contributions do not seem to affect the classification performance much, being in the same region as those for their MDS-condition which is implemented here as well: There are no really significant improvements that are not given here, giving no reasong to assume that their performance should be superior.

Not having some improvements of [3] The **Ortho** condition of [3] actually does significantly outperform the base algorithm for many configurations. For Foursquare, their performance comes very close to mine, whereas their GeoNames-performances are a lot worse than mine. Apart from that, if the code uploaded by [3] is really the basis for their implementation, ther are strong reasons to doubt what they claim to do and what they actually do really matches. A quick inspection of their uploaded source code[5] revelaed for example they take the kappa-score from on the raw predictions, not on the rank like [1] described and like we do (see Table B.1) and also do not apper to weight the kappa-scores. Apart from observations like this, it is hard to say more about their code, because the two files uploaded by them are not the whole algorithm and also depend on loading many files that are not in the repository, and also none of the evaluation on basis of decision tree performance is in the repository.

Using the best configuration Another difference appears to be that we looked for the best configuration *for this particular task*, which is a different one for each combination of dataset × DT-depth × classification-target. It appears from their description that [1–3] did hyperparameter-tuning before, using another possibly subjective metric, and then decided on one (or rather four, see Table C.2) configuration that was not optimized for the dataset × DT-depth × classification-target. It should be noted that in this work, the algorithm is also not optimized for that task, but only the best of the 80 different parameter-combinations that were executed is used respectively. At the same time however, some way to find a hyperparameter-configuration has to be used, and it is unlikely that [1–3] chose the worst configuration. Table 4.2 displays robust results of a parameter-configuration that proved good on average. As these results, however, also

[4] `https://www.cs.cf.ac.uk/semanticspaces/`

[5] `https://github.com/rana-alshaikh/Hierarchical_Linear_Disentanglement/blob/master/Hierarchical_Linear_D4.py#L485-L486`

outperform those of [1–3] significantly, choosing the best configuration seems to play only a minor role for the performance.

Weighted Average of the individual classifiers The considered scores for the 1vsRest-condition of this work are calculated from the scores of the individual per-class classifiers both with uniform weighting per class (bottom two rows of Table 4.2) and also with class weights inversely proportional to class size (middle two rows). Clearly, the condition that weights the indiviudal scores leads to better results. Weighting the score is a reasonable assumption, given that the individual class frequencies are very imbalanced (see Figure C.1). Unfortunately, [1–3] do not explicitly share if they calculated weighted class scores as well. If we assume for now that [1–3] did report unweighted scores and thus disregard the condition where class-scores are weighted (see bold scores in Table 4.2 or last column of Table C.2)), our results are still comparable with those of [1–3] and especially in the case of GeoNames-labels a lot closer to those reported in the literature. So while we hereby argue that weighting the scores makes sense, even if that is not the case our results are still acceptable. Again it should be stressed that there are only really few and imbalanced labels for this dataset in general, making the statistical power of these results very small.

Improvements that we do have We do not have many differences in hyperparameters or algorithm-components than [1–3] do, but there some. For example using tf-idf as quantificiation instead of PPMI: Close inspection of Table 4.10 shows that often, the tf-idf results are superior to the PPMI-results, and sometimes the combination of using tf-idf as quantification and tf-idf as dtm-quantification is good. This may lead to two conclusions: Either tf-idf is just better than PPMI under certain conditions, or just that the fact that more different results generated here just increased the statistical chance that good results were among the generated ones.

Conclusion

Even though this work did not do much beyond [1–3], maybe there were some small things that were done here that gave an edge, such as trying out different or just more hyperparameter-combinations. Maybe the scores here were calculated differently than in [1–3], but even if that were the case the results generated here are still comparable. Maybe our implementaion had errors, maybe those of [1–3] had, but in any case as the dataset is so small exact results do not seem incredibly informative anyway. Most importantly the comparison was performed to check if this implementation is working correctly, and the evidence for that appears very strong.

5.1.2. Results for educational resources

One of our two research questions was to figure out if the methodology works for our domain. So now that we have established that the implementation seems to work on other datasets, we finally look how the algorithm copes with the Siddata-dataset.

Quantifiable dataset differences

When describing the datasets in Section 3.1, we noticed that they are quite different. An important difference from our to the originally used datasets is, that in our dataset

the relationship between how relevant a concept is for an entity and how often its words occur in the respective BoW is not given. Furthermore, the Siddata-dataset contains far less words per entity: the median number of unique words per description is three orders of magnitude smaller compared to the placetypes-dataset (see Table 3.2). Even though the number of entities in the dataset is higher, in sum it contains substantially less unique words. The difference is very prominent in relation to the dataset size (indicated by the last two columns). Because of this, for most of the n-grams that serve as candidate-terms in the subsequent classification the positive class (*entities that contain the phrase*) is far smaller than the negative class.

Derrac & Schockaert [1] extracted roughly 20 000 candidates for the movies- and placetypes-datasets. If the same number of candidates were to be extracted in our case, more than half of them would occur in less than 25 descriptions, such that the positive class for the corresponding classification problem contains only $\frac{24}{26346} \approx 0.09\%$ of the samples. It seems unlikely that even class weighting can make up for that, yielding a bad classification. Because of this, it is justified to consider less entities. To improve the ratio of classes, it was accordingly decided to only consider entities of at least 80 words (see Table 3.1), which yielded the final number of 11 601 considered entities.

When discussing the dataset differences in Section 4.2, we assumed that the dataset differences will likely lead to less candidates being extracted. This is confirmed by the results: Figure 4.1 shows that the number of candidates that apply to each entity is exponentially decreasing. The score represents a low *faithfulness* in the representational capacity for many of the produced candidates. This indicates that there is a much variability in the dataset that is not explained by any of the extracted words. A consequence of this is that mapping this space onto a limited number of extracted directions will lead to a loss of information which does not model the full latent information.

Despite the small number of extracted candidates however, a sample run with 200 dimensions still yielded 5016 phrases with $\kappa \geq 0.1$ ($T^{0.1}$) and 1008 with $\kappa \geq 0.5$ ($T^{0.5}$), which is enough for the algorithm, considering that for a 200-dimensional embedding only 400 values with $\kappa \geq 0.5$ would be necessary. However, there are far less ones in $T^{0.1}$ compared to Derrac & Schockaert [1], meaning the resulting clusters are considerably smaller.

This, however, is not necessarily a sign of bad performance of the algorithm: The number of cluster-elements that [1] for the 200-dimensional CS for the placetypes dataset (see Table 4.3) is 21 819. Considering that they considered number of candidate is 21 833, the threshold does not meaningfully reduce the number of considered words. Accordingly, in this dataset all extracted words (which are all words with $df \geq 50$) are considered in the final embedding. This leads to high amounts of noise, i. e. a bad modelling of the actual latent topics.

Increasing the threshold a candidate to be considered a *faithful* representation also does not help: Consider the **Sum** column of Table 4.3. The first two rows of Table 4.3 display sample results of Derrac & Schockaert's [1] original implementation for the domains of movies and placetypes as it was uploaded by the authors. Let us consider the Sum column, which indicates how many unique terms have been identified and used as *most*

important feature direction among all different uploaded conditions. The authors uploaded their results for embeddings of the dimensionality 200, 100, 50 and 20, in each of which the number of extracted cluster centers was 2*ndim. Considering this, the minimal number of cluster centers that could be extracted among all their uploaded results is 400. The worst case would be given, if the sets of *salient* terms for each of these runs would be completely mutually exclusive. In that case, not a single term that was considered *salient* by one of these results would considered salient by any other configuration, such that the sum of unique terms extracted among all combinations is 400+200+100+40=740. This can be seen as a measure of *Robustness* of the algorithm. If different parameter-combinations or just different initial random number generator results have a high impact on the generated results, the algorithm is not robust. In the case of our algorithm this shows in different extracted candidate terms. For the placetypes-dataset, 697 different semantic directions are found. Considering the condition $\kappa \geq 0.1$, 21 832 of the 21 833 candidates were considered salient among the runs. This indicates much noise in the dataset that obfuscates the latent information.

These observations lead us to the conclusion, that extracting less candidates may better capture the semantic content of the dataset. The disadvantage is that the resulting embedding captures less variance of the original dataset, however, those directions that are extracted show increased robustness. This is also indicated by the lower number of uniquely extracted candidates summed over all run-configurations in Table 4.3.

When discussing the dataset, we already theorized that only keeping those entites for which a classifier can successfully predict its faculty may help to increase dataset quality. That turned out to be not necessary but would still be a good future research opportunity. Another possibility that could have been considered in the case of low performances is to use only the 1500 with the longest descriptions, bringing its distribution closer to the placetypes-dataset (but not changing the properties).

In sum, our previous hypothesis that the different dataset statistics leads to different conditions for the algorithm seems confirmed by the intermediate results. On the other hand, the final classification performances (Table 4.5) are proof some important information of the dataset is captured regardless. In fact, not only are enough candiates extracted by the algorithm, but the results even indicate less sensibility for different hyperparameters for our dataset compared to the results of [1] for the placetypes-dataset. Regarding the algorithm, our results indicate that the methodology is robust and does not only work for datasets with the aforementioned properties.

Classification results

Comparing the t-SNE-embeddings of the classification problem associated with the Siddata-dataset (Figure 4.2) with the one of the placetypes-dataset (Figure C.1) indicates that the former seems to be more prominent in the data. This saliency of the faculty in the representation of the samples is further confirmed by the fact that both our implementation of BERT and even representation relying on only three dimensions achieve reasonable performances on the data.

We should keep in mind that the problem appears to be comparably easy when evaluating our performance. Despite this, our acccuracies are suprisingly good: Our algorithm robustly achieved 81.4% accuracy with depth one trees, which is comparable to the accuracy for BERT (85.19%), and by far outperforms the 3D-embedding (64.3% weighted accuracy). This indicates that our method indeed finds terms that accurately predict the faculty among its salient directions. A classifier that that uses only three dimensions is already better than BERT, and the unbounded one has 94.3% classification accuracy. This stands in contrast to the results of [2], who report that their depth-1 trees achieved the best overall performance. In contrast to the placetypes-dataset, unbound trees for this dataset appear not to be overfitting. This can be explained by less noise and general variance in our dataset. Another interesting observation from these results is that the variance is on average comparably low, indicating robustness.

Unbounded DTs are actually only an answer of the question if it can recover at least one property from the dimensions, not how important one or any of the dimensions are. So their good performance is only a measure of the question if the information about the faculty is not lost through the embedding.

Interestingly, classifying all faculties at once performed a lot worse in case of the Siddata-dataset (Table 4.4). This difference in performance is a lot more prominent than it was for the placetypes-dataset. Considering this, a fairer comparison to BERT would have be to also split the dataset into several *1vsRest* problems.

When looking at the individual faculties, we see that for the faculty *Humanwissenschaften*, depth-1-trees perform a lot worse than for the other faculties, but also shows a large standard deviation . This indicates that the classification performance for this faculty strongly depends on what ends up in the train set, leaving the interpretation that the individual descriptions of courses belonging to this faculty have a higher degree of variance in their descriptions. This is an interesting result, as we would have expected other faculties that contain many different courses of study to have a higher variance.

While these quantitative results look very good, it is important to stress that they only refer to how good the faculty can be detected from the semantic directions, which is only one (apparently very prominent) property of the data. Evaluating the algorithm this way is in line with the literature [1–3], but does not test everything the algorithm does or how it may perform with respect to other human concepts among the data. In our evaluation, we also evaluated the algorithm on its performance when classifying the first level of the DDC as detected by SidBERT with similar performances.[6]. As, however, the categorization system of that shows high similarity to the categories made up by the faculties, the results for these are not printed in this work. Other avenues may include to try to detect the *difficulty* of a course on basis of average grades, the *interdisciplinarieness* on basis of the number of students from other faculties attending it, or the *effort* required for it with labels generated from the amount of ECTS a course provides. Unfortunately, metrics are not present in the current dataset. Ultimately, the best way to objectively

[6] `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/notebooks/analyse_results/siddata/decisiontrees_bestconfig.ipynb`

quantify the quality of the directions remains to perform studies with human subjects evaluating their subjective appeal.

Dataset Quality

Now that we have established that we have reason to assume that our algorithm is good, let us interpret some results with respect to the dataset. We have extensively discussed our dataset with respect to the frequencies of words, let us now try to see if we can speculate about some more of its properties.

Among others, we are interested if any important information gets lost when using only the extracted semantic directions to embed the entities. The results for unbounded decision trees (Table 4.5) which achieve 94.3% weighted accuracies indicate that at least regarding information about the faculty seems to not get lost. This indicates again that our directions are useful to express the variance in the data, considering that the results for the faculty are not trained on but are only a side effect of the extracted semantic directions.

Another method conducted to test how much variance in the dataset is explained by the semantic directions was to check how many dimensions and what amount of discretisaton was required such that multiple entities fall onto the same position in the left-over vector space. Table 4.6 shows how many dimensions are required such that enough entities are unambiguously defined. The results indicate that without much discretisation of the space, a small number of directions is enough such that no duplicate positions will be taken. This indicates that much of the variability in the original data can be explained by a small number of dimensions. Considering however that these are random directions, it also indicates a high correlation/linear dependency of the dimensions that are left over, otherwise this would only hold if the leftover directions happen to be the good ones.

It is, however, much more interesting to analyse which of the entities did become duplicates through this. Table 5.1 shows a sample where three out of four entities that landed on the same position refer to the same course of different years. Results like this indicate that all results concerning ambiguity of samples must be taken with a grain of salt, as the dataset even in its cleaned form seems to still contain many duplicates.

Course Name	**Leftover Dimensions**		
	strukturierten	internationales	1950er
!! FÄLLT AUS !! Anna Seghers: Das siebte Kreuz (1942)	2	15	18
Anna Seghers: Das siebte Kreuz (1942)	2	15	18
Anna Seghers: Das siebte Kreuz (1942) (NDL 2)	2	15	18
Friedrich Hebbel: Agnes Bernauer (1852)	2	15	18

Table 5.1.: Sample courses falling onto the same embedding after discretisation.

Qualitative analysis of the extracted dimensions

Table 4.7 shows the extracted feature directions that, according to a DT classification, best predict a faculty. Many of the extracted features are subjectively very appealing in encoding the faculties - in some cases even extracting the exact faculty name. Figure 4.4 displays the results from another hyperparameter-combination, which are similarly con-

vincing. In a recommendation system, many of these directions would make sense as features semantically describing a course. Table 4.8 shows some exemplary terms that make up a cluster. These look intuitively appealing as well.

When looking at the examples per faculty, we observe mixed results. On the one hand, e. g. *erziehungswissenschaft* is perfeclty on point, on the other hand the extracted terms for faculties for *Physik* and *Biologie/Chemie* did not look convincing in a single run. One possible explanation may be low class frequency. Both mentioned faculties are among the smaller classes, but so are *Mathematik/Informatik* and *Wirtschaftswissenschaften*, both of which are generally classified really well.

Do bad performances correlate with bad feature names? In almost all results, the extracted dimensions for the faculties *Physik* and *Biologie/Chemie* appeared the worst. The results for these faculties show the highest variance and do not make much intuitive sense. The top extracted directions (Table 4.7) show that the dimensions are good, explaining why even the performance of depth-one-DTs is good. Quantitatively, we observed the worst performances for the faculties *Humanwissenschaften*, followed by *Physik* and *Biologie/Chemie*. Looking at Table 4.7 shows reasonable results for all faculties except for *Biologie/Chemie* and *Physik*. That is interesting: Even though for *Humanwissenschaften* the performance is generally not too good, the dimensions we extracted from it are plausible.

NDim	**50**		**200**	
	0.1	**0.5**	**0.1**	**0.5**
computer	24	0	4	11
sport	10	12	4	11
recht	27	0	11	4
musik	17	12	7	8
management	25	0	9	6
literatur	0	0	0	0
sprache	22	0	9	6
psychologie	23	2	3	12
wirtschaft	25	0	13	2
geographie	28	0	10	3
schule	17	8	7	6
kultur	22	0	5	8
wissenschaft	17	0	14	0
sport	10	12	4	11
N	38	38	15	15

Table 5.2.: Number of appealing phrases found among the directions. Results show the number of parameter-configurations in which the respective word was extracted as direction. Last row indicates how the how many configurations are considered for each column.

Are intuitively appealing phrases among the semantic directions? In Section 3.4.2, we decided on some terms that would make intuitive sense as semantic directions. The results from Table 5.2 show that many of these were actually used as candidates robustly among many different parameter-combinations. This is a very promising result, especially considering that the actual document frequencies for these terms are comparably

low, making the result even more astonishing and indicating robustness by how many different configurations found these teerms to be significant.

computer	1.0%	wissenschaft	3.3%	musik	2.27%
mathematik	0.8%	recht	4.16%	sport	1.03%
mathe	0.0%				

Some of these phrases even regularly end up as a deciding factor to detect one of the faculties in the decision trees, such as *computer* for *Mathe/Informatik*, *psychologie* for *Humanwissenschaften*, *Schule* for *Erziehungs/Kulturwissenschaften* or (interestingly) *wirtschaft* for *Sozialwissenschaften*. This was even much more pronunced when also counting if the phrase was part of the word, such as *recht* being in the detected dimension *wirtschaftsrecht*. All extracted ones are: *schuldrecht, strafrecht, deutsch recht, offentliches recht, grundrechte, urheberrecht, steuerrecht, strafrecht, deutsch recht, recht, grundrechte*.

Are embeddings of known similar entities close?

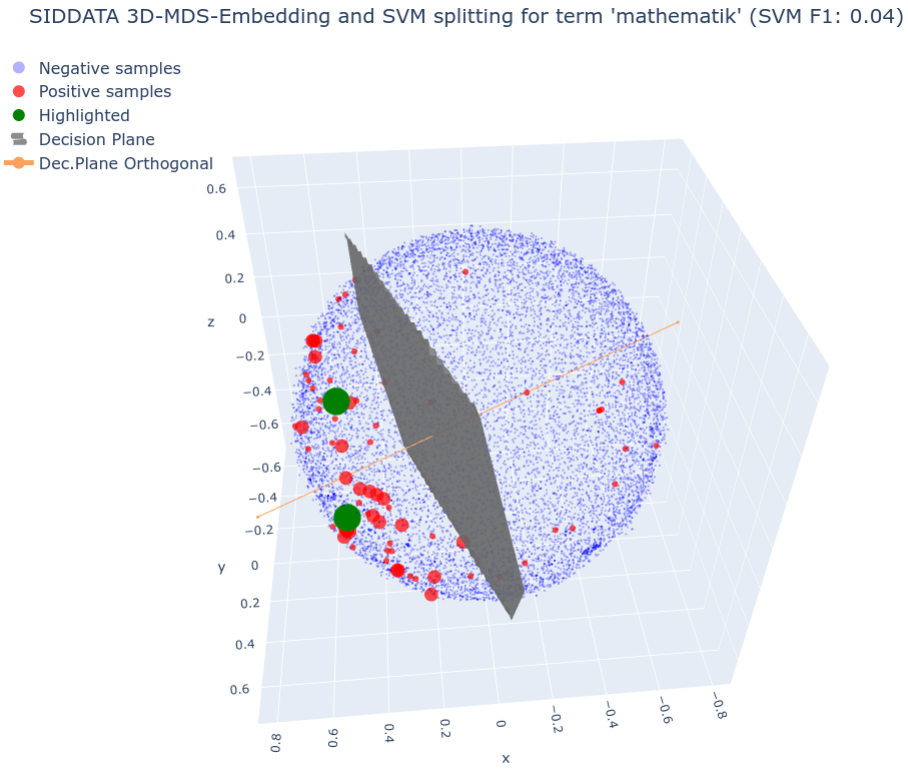

Figure 5.1.: 3D-Plot with an SVM for the term *mathematik*, also highlighting the courses *Informatik A* and *Informatik B*. Negative samples are hidden for better visibility, and entities that contain the word more often than the 75$^{\text{th}}$ percentile have bigger markers.

Figure 5.1 displays a 3D-Embedding for courses, splitting courses which contain the term "*mathematik*" from those that do not, also hightlighting the courses *Informatik A* and *Informatik B*. Both courses are close to each other for both the *normalized angular*

distance and also euclidean distance.[7] Further, even though neither course description contains the word *mathematik*, both courses would receive high values for this feature, indicated by the distance to the decision plane.[8]

Are top ranking courses for the directions convincing?

To further analyse the quality of the results, we are interested in the top courses of those directions that best represent the different faculties. Table 5.3 displays the most prototypical examples for each of those, which is directly derived from the rankings induced by the classifier.

Faculty	Top Direction	Top Course for Direction
Erziehungs-/Kulturw.	erziehungs-wissenschaft	BA GM 3.2: Der Kinder Zukunft aus elementarpädagogischer Sicht
Rechtswissenschaften	schuldrecht	Strafprozessuales Ermittlungsverfahren, [...]
Wirtschaftsw.	fallstudie	WIWI-B-18001-WI: Management Support Systems B I (BI-Praktikum SAP/BW)
Kultur-/Geow.	stadtgeographie	Mittelseminar/Angewandtes Seminar: Geographische Handelsforschung [...]
Mathem./Informatik	mechanik	Technische Mechanik IV für Maschinenbau
Sprach-/Literaturw.	deutschen literatur	Deutsch - diachron: Historische Linguistik und Sprachwandel in der Gegenwart [...]
Humanwissenschaften	gehirn	Willensfreiheit und Hirnforschung [...]
Physik	klar	B1-B2 Schwedisch Erweiterungskurs: Schweden - das Land und die Menschen (DIGITAL)
Biologie/Chemie	stattgefunden	Transformation wohlfahrtsstaatlicher Regime in Europa: Aktuelle Forschungskontroversen
Sozialwissenschaften	parteien	Modul Vergleichende Politikwissenschaft I: Westliche Regierungssysteme im Vergleich

Table 5.3.: Highest ranking courses per feature that best predicts the faculty.

Again we observe that many of these make intuitive sense, but *Physik* and *Bio/Chemie* cannot be satisfactorily captured by our dimensions. *Mechanik* was an unfortunate choice to encode *Mathematik/Informatik*, however the corresponding course for the *direction* definitely seems resonable.

5.1.3. Hyperparameters results

After much preliminary elimination of other techniques, we decided on 165 different hyperparameter-combinations for the final run, which are what was decided on in after much preliminary elimination of other techniques.[9] For example, prior experiments have shown that the *reclassify* algorithm to find cluster directions outperforms the one used in [1], which is why the latter was not considered in the final evaluation, to counteract a combinatorical explosion of hyperparameters.[10]

Classifier-Scoring-Methods Let us first consider if the choice of the exact evaluation method to quantify the classifier performance is relevant. For that, we look which score leads to the best result, and if the results differ drastically. Table C.5 clearly indicates that the algorithm is very susceptible for the exact scoring method used. It was to be expected that comparing raw counts to rankings performs worst (column *c2r+*), and that

[7] Normalized Angular Distance: Average 1.00, Info A and Info B: 0.37 (Percentile 8.2), Euclidean Distance: Average 0.88, Info A and Info B: 0.39 (Percentile: 8.9)

[8] Both statements are better visible in the interactive version of the plot

[9] An exception to this are of course the results for three dimensional spaces, which are not competitive and only generated because they are more intuitive and presentable.

[10] With regards to this specific parameter, there is however qualitativ evidence given in Table C.3.

was confirmed. Besides this, it is also interesting how big the difference between *rtr2+min* and *r2r+max* is, considering that they only differ in how they deal with duplicates. It was suprising that *digitizing* the values worked at all. We see that the only scoring-methods that produced enough (2*#Dims) cluster-centers are those that only considered those terms that occur at least once, which makes sense given that otherwise the scoring was too much influenced by the number of elements in the positive class.

To get a better grip on the algorithm's robustness with respect to the exact scorer choice, let us now also consider their overlap as printed in Table 4.9.

Some things we see include:

- binary kappa and F1-score have a high overlap (kappa being a bit stricter)
- comparing rankings of only the positive values and their digitized versions leads to the exact same results
- all of r2r+min/dig+2, kc2r+ are in b2b
- all of the onlypos-statistics are completely in the respective kappa bin2bin

Furthermore, looking at the individual scoring methods also provides insight with regards to the question if the different nature of our dataset needs to be accounted for: As established, a very important difference is that more relevant words do not occur more often in the Siddata-dataset. This makes it a reasonable assumption that scorers that evaluate the *rankings* should perform worse: If it is not the case that important words that are very relevant to an entity necessarily occur more often in its associated text, comparing its ranking should perform worse than considering the classification as a binary problem.

We can test that by comparing the overlap of features that achieve high kappa-scores with those that achive high scores of binary metric such as *accuracy, precision, recall* and *F1*. Table 4.9 indicates first that the binary metrics are a lot less strict than the kappa-scores: All those terms which exceeded the threshold of 0.5 for one of the kappa-scores also did so for accuracy and recall. For *precision* and *F1*, however, this is not the case. More evidence is that the kappascores are a lot worse on count than they are on quantificiations. Furthermore, the kappascores are a tiny bit better on lemmatization than without it, providing even more evidence, as it is to be expected that this configuration has a slighly higher word count.

Other Hyperparameters Note that the results reported in Table C.5 were considered w. r. t. deciding which ones to take for the final runs. This refers only to the decision which scoring methods to use but also about the hyperparameters up to that point - if not at least 2*ndim candidates for clustercenters are generated, the config is useless. In fact, it can be assumed that more extracted values are generally better, because if only the minimal number is generated all of them will become cluster center no matter their degree of collinearity, which makes the resulting space useless. With that in mind, Table C.5 suggests that a parameter-config with 200 dimensions that *tf-idf* will tend to produce most dissimilar candidate directions.

In general, the results for different hyperparameters in terms of classifier performance (Table 4.10) indicate some trends:

For one, it definitey shows that comparing the raw count with these rankings performs worst, indicating that while the wording in Derrac & Schockaert's [1] was ambiguous, they likely talked about comparing the respective ranks. The second predictor of below-average scores is the usage of a three-dimensional embedding. This was to be expected, and given the low expressive power exptected from a three-dimensional embedding, the observed scores are even surprisingly high.

Comparing PPMI and tf-idf as scoring methods does not show any clear trends, however we can also not find good reasons to prefer the former over the latter. Given that calculation of PPMI-scores relies on huge matrix multiplications which are computationally inefficient, we would suggest to use the well-known well-optimized tf-idf score instead. Another insteresting observation is that scoring the Candidate-Matrix differently than frequency matrix of the texts is competitive with using equal scorings for both. Also, even using the raw count for the candidate-matrix did not perform very badly, serving as our final piece of evidence to see the robustness of the algorithm for datasets like ours.

Besides these observations, the other hyperparameters did not seem to affect classification performances much. In the printed table, the highest score is yielded by a 200-dimensional embedding with tf-idf encoding. However the results slightly varied for different random seeds due to differing elements ending up in the train- and test-set, leading to slightly different results with another hyperparameter-configuration ending up slighly better than the one printed in this thesis.

5.1.4. Did we achieve the thesis goal?

Let us now take a step back and answer the question if the results indicate that we achieved the aims of this work. For one, there is certainly more work to do to increase robustness of the results in the sense that an unambiguous name for the resulting semantic direction comes up without being affected much by precise hyperparameter choices or random seeds. The fact that we have tested only by comparing how good the classification can be predicted is also a drawback, but a human study would be clearly outside the scope of this work.

Our results and analysis reach all expectations set in Section 1.3.1: We have extensively analysed the difference between the Siddata-dataset and the originally used dataset. Though many differences between the datasets have been found, it has been shown that the algorithm can make sense of the dataset regardless. All results printed in this work are open-source and the original notebooks are freely available to prove that we did not cherry-pick any results for the analysis performed of the previous section. The semantic directions we have found allow to predict a courses' faculty, which encodes an important feature for the recommendation of educational resources. Our performances to predict the faculty even outperforms a classification relying on BERT.

5.2. General Algorithm

After having presented and meticulously implemented the algorithm of Derrac & Schockaert [1], we are familiar enough with the relevant theory and practice, allowing us to now critically reflect on the algorithm in general.

5.2.1. Algorithm idea

When first looking the algorithm of Derrac & Schockaert [1], it appears astonishingly specific. However after having read Gärdenfors' book Gärdenfors [18], the idea for their algorithm is self-evident: In it, Gärdenfors suggests that conceptual spaces can be generated from high-dimensional sensory input by using MDS to project the original data into a Euclidean space to do geometric reasoning in that space. The book has an entire chapter on computational aspects, in which the author discusses vector space models, dimensionality reduction techniques an ANN architectures for different levels of human conceptualization. According to that, MDS is especially good at dealing with pairwise distances judgements from a subject's perception, to create a more *economic* representation for *phenonemal* CSs [18, p. 221]. Derrac & Schockaert [1] did not create the space from sensory data but from text corpora, where the distance of two texts can be measured by the words they share. Their algorithm reasonably combines the idea to generate CSs with the steps for a classical NLP pipeline as described by [41] (Section 2.5.1) Given that, the core contribution of [1] mainly lies in the idea that the *faithfulness* of a potential direction for the resulting semantic space can be assessed by the performance of the corresponding decision problem.

Measuring the Faithfulness of directions It seems reasonable that this assumption holds for the datasets originally considered by Derrac & Schockaert [1]. As extensively discussed, when concatenating reviews of movies or tags describing pictures of places it is natural that words that describe a salient feature of the respective entity occur more often. The Siddata-dataset consists of short descriptions without this property, where most words have a relative document-freqeuncy of less than a percent. However, the technique of comparing the ranking induced by the classification with the score of the words still yielded enough results, and the directions seem to consist of interpretable properties. This surprisingly confirmed the robustness of the algorithm in that regard.

Requiring MDS

Derrac & Schockaert [1] explicitly state that MDS is the best dimensionality reduction technique for their algorithm, as it is one of the few ones that result in a metric space. In their paper, they describe SVD (the mathematical algorithm behind LSA) as a popular technique for dimensionality reduction, but further state that "SVD produces a representation in which entities correspond to vectors, which should be compared in terms of cosine similarity rather than Euclidean distance. [...] However, we can expect that spatial relations such as betweenness and parallelism [...] are not meaningful in the representations derived from SVD" [1, p. 14]. These relationships are required for semantic classifiers which mimic analogical and betweeness-based reasoning, which they demonstrate to work for all of their domains. However, this space does not have semantic directions. The *final* feature-based representation of the entities is reached by ranking each entity for each of the feature directions and creating new vectors from these ranks. As acknowledged by Derrac & Schockaert [1, p. 22], the feature vectors are not orthogonal, not linearly independent and only of ordinal scale level, withhout meaningful distances. Derrac & Schockaert [1] create semantic classifiers both for the *intermediate* space with meaningful

distances (geometric betweeness- or parallelism-based classifiers) as well as for feature-based representation (a-fortiori-classifiers, see Section 2.3.2). However, Ager *et al.* [2] and Alshaikh *et al.* [3] are only interested in the latter space and its resulting feature axes. If that is the case however, it becomes irrelevant if the intermediate space is metric or not, which enables for other algorithms to be used in that step. As stated in Section 2.5.1, LSA may be the better choice as it explicitly detects latent topics in descriptions instead of relying on words that are explicitly mentioned. This may also lead to other desirable properties such as comparability of documents and phrases.

If for the explainable classifiers the relevant space is the one with semantic directions, geometric properties of the intermediate space are irrelevant. Instead one should try to generate a space that has both a useful metric *and* interpretable directions. Depending on what the authors consider to be the end result of their algorithm, only one of these necessary requirements for conceptual spaces is fulfilled. So if their end-result is the intermediate space, that is nothing more than a normal VSM with some interesting but long-term useless geometric properties. Instead, another possibility may be to calculating a new orthonormal basis on the coordinate system. The next step would then be to enforce orthogonality of the semantic directions as good as possible, and using linear algebra for a change of basis for the entites, such that not only their ranks but their exact position for each semantic direction (axis) is relevant. Subsequently, one could use techniques like Principal Component Analysis to decorrelate the directions. In this way, one would obtain vector without a name, which however could again be found with techniques that rely on cosine distances such as LSA.

Ager *et al.* [2] and Alshaikh *et al.* [3] both do not this interim space and only use re-embedded one, but retain the use of MDS regardless. To our best understanding, they have no reason to do so, giving impression that they read to use MDS in Gärdenfors' book and then forgot that their final re-embedding step makes that irrelevant.

Using classical techniques In fairness, [2, 3] both experiment with modern neural embeddings for the entities such as averaged Word2Vec or Doc2Vec. Both authors report that this decreased performance (see Table C.2) compared to their MDS-condition.[11] However, while they use neural embeddings, they do not adjust any of the later steps to regard for that: they just replaced the VSM generated classically in the first steps of the algorithm with a neural embedding, such as averaged GloVe embeddings [50] of the words in the text. On that they ran the exact original algorithm of creating a frequency matrix from the BoW and using a linear classifier to get the direction.

They do not consider looking for latent topics or make use of the algebraic properties that are given for these embeddings (see Equation 1.1). A possible avenue could be to look for shared vector components of entity-embeddings and candidate-word-embeddings. Instead, they still rely on the usage of linear classifiers that split according to frequency matrices. We will look more detailed into that soon, but this makes much more sense for *points* in Euclidean spaces than it does for *vector* embeddings where the space is built up from a cosine-distance-similarity-based objectives (see Section 2.5.1). Instead of that,

[11] Even though Alshaikh *et al.* [3] state in their paper that relying on better embeddings such as BERT [15] may lead to better results.

it seems the better idea to take advantage of dealing with vectors, such as the fact that they are inherently directional already. As discussed in Section 2.5.1, it seems that the concepts of LSI to compare the similarity of documents and candidate feature directions seems more appropriate. This has added benefit that not only words literally occuring in the respective texts can be considered, which among others counters the previously stated problem that the used classifiers deal with heavy class imbalance as well accounting for polysemy and synonymy.

5.2.2. Does the algorithm actually produce a Conceptual Space?

Section 2.3 explained what a conceptual space is, before introducing Derrac & Schockaert's [1] algorithm to automatically induce them. However, their algorithm only approximates CSs and does model some parts of the definition. A first difficulty with the algortihm is, that it needs a clearly defined domain from the start. It takes a corpus of texts and embeds each of these into a single high-dimensional vector space. If not all texts in the corpus are from a single domain, due to the similarity-based vector space generation, outliers will greatly affect the embedding of all entities (as [2] discusses at length). This sounds irrelevant in practice, but the problem is that it is impossible to clearly define what a domain is. There is the set of place types, but this domain consists of various subdomains, and some concepts apply only to a specific subset of entities. The described algorithm cannot figure out such subdomains. This issue is partially addressed by the works of Alshaikh *et al.* [3, 35, 36] which elaborate on the idea of subdomains. As they state, *"When representing a particular entity in a conceptual space, we need to specify which domains it belongs to, and for each of these domains we need to provide a corresponding vector."* [3][12]

Also it is important to be aware of the difference between what [1–3] and this work consider a domain (the set of movies, places or courses), and the definition of domain as used by Gärdenfors [18]. According to the latter, a domain is a low-dimensional set of correlated properties, such as *the color domain* consisting of *hue, saturation* and *value*. This also points out another difference of the original CS definition and the definition used here: Conceptual spaces describe an entity through several uncorrelated low-dimensional vector spaces, not a single one with several dozen dimensions. As [2] puts it more humbly, *"The idea of learning semantic spaces with accurate feature directions can be seen as a first step towards methods for learning conceptual space representations from data [...]"*. Again it is referred to the works of Alshaikh *et al.* [3, 35, 36] which alleviate this by iteratively finding disentangled low-dimensional feature spaces.

As discussed earlier, the algorithm of Derrac & Schockaert [1] first embeds the entities into a Euclidean space where the concepts of betweeness and parallelism make sense, and subsequently create a feature-based representation that bases on an entity's rank w. r. t. several human-interpretable features. The final embedding is only of ordinal scale level and thus unable to model degrees of similarities. In other words, the algorithm

[12] Their improvement to the work of Derrac & Schockaert [1] is many to introduce the concept of subconcepts to the algorithm by hierarchically disentangling the generated space into subfeatures that only exist if certain top-level features are given, such as encoding the *political orientation* of Organizations only for those have a high degree of being *political* in general.

produces *either* a space with a euclidean metric, *or* one with interpretable directions, but no space that has *both properties*. As both are necessary conditions for a conceptual space, Derrac & Schockaert's [1] algorithm at no point generates something that resembles a CS. It remains unclear to us why they only consider the ranking of the entites regarding the feature axis instead of their distance which may retain relations of distances, for example by applying an arithmatic change of basis for the coordinate system (linear transformation). The way their algorithm works, the final space only has ordinal scale level and linearly dependent (correlated) dimensions. An interesting research avenue is to figure out what properites the space they have, and if small changes to the final algorithm step (such as PCA to decorrelate dimensions or not only taking the rank) could help in retaining the euclidean or at least another usable metric.

Points instead of Regions

Another important difference between the resulting space and CSs is that we are dealing with points instead of regions. Advantages of doing that include that it allows to distinguish *protypical* examples from borderline cases [18] and straight-forward application of ontological relations through the RCC (see Section 2.3.2). Derrac & Schockaert [1], however, drop this assumption and work with vectors instead of regions, claiming that this is a "coarse-grained approximations of conceptual spaces, where points correspond to fine-grained categories instead of specific instances, while convex regions are used to model higher-level categories" [1, p. 8]. Despite that, they never address it again, so in the end they stick with points. This breaks with one of the key concepts from CSs and also renders it impossible to simulate any of the ontological relations with the resulting spaces, which Gärdenfors considered their most relevant practical application [23].

However, when elaborating on the idea that conceptual spaces can be induced using Kohonen-Nets (Subsection 2.4), Gärdenfors himself claims that mapping regions of the original space to point-embeddings in the CS is an *advantage*, because it resembles *generalization*. Another aspect to consider is whether the entities as considered by [1–3] actually resemble what Gärdenfors originally considered an entity. The entities in the used datasets of educational resources, placetypes or movies actually are specific instances instead of general *concepts*. Instances in a conceptual space are correctly modelled as points - one could say that regions denote *types*, with the individual points corresponding to their *tokens*. Considering that we have only one instance per entity, we are dealing with types. Concepts (Regions in a CS) could accordingly be induced as the set of multiple tokens. A practical implementation could, for example, model the concept of *introductory computer science courses* as a convex region spanned by all of the instances it contains. Erk [56] propose an algorithm that creates such a region with varying variances per dimension from a set of prototypical instances. It should be noted, however, that learning boundaries for such regions requires much more data [1] and the aforementioned reasoning on regions is computationally very complex [57].

Vectors or Points The authors explicitly claim that they are dealing with points in a Euclidean space, which should be compared in terms of Euclidean distance [1, p. 14] instead of cosine distance. Despite this, in the merge-canidates step of their algorithm

(Section 3.2.1), they compare the candidate feature directions using the cosine distance of their orthogonals. As they require similarity of directions and not of positions, this approach seems plausible. However, these vectors do not have their origin in the coordinate base but in the position where they cut across the SVM's decision surface: A SVM is described by the vector of its orthogonal intercept, a scalar describing where the decision hyperplane crosses it. This means that one is dealing with *affine frames* (which are described by basis and origin) instead of vector spaces. However, vectors in affine spaces cannot be compared solely by the angle between them[13]. The way their algorithm is described, their merging of feature direction disregards the origin. Consider the following example: The SVMs for two candidates have exactly the same orthogonal vector, but different intercepts. There might be samples between the two decision surfaces, which are classified towards the positive class by the one classifier, and towards the negative by the other. If they classify samples differently, they must express different concepts. When only accounting for the direction of the orthogonal, such information gets lost.

5.2.3. Outlook

There are also techniques that extend the algorithm of Derrac & Schockaert [1]: Alshaikh *et al.* [35] take a vector space embedding and decompose it to several low-dimensional spaces, such that it corresponds more closely to the definition of a CS which are split into multiple domain-specific spaces of low dimension. For that, they take the spaces from [1] to then cluster their features by domain and iteratively remove these groups to create multiple subspaces, while ensuring that Word2Vec embeddings close to those of the removed ones are disregarded for future features.

Alshaikh *et al.* [36] want to get rid of MDS with its quadratic space complexity and also write a completely new, unsupervised ANN algorithm based on GloVe embeddings [50]. In it, they learn domain-specific embeddings from the BoW and like [1] use classification, splitting entities that contain one of the verbatim candidates vs. those that do not. They train an ANN on this while also punishing close embeddings, similar to their previous work [35].

5.3. Architecture

As one of the thesis goals asked for a good and scalable architecture, we will also evaluate whether the implementation fulfills the set criteria (Section 1.3.1) and if the aspects we considered for a qualitative software and sustainable data analysis (Section 2.1.2) are fulfilled.

First of all, we wanted to show that this implementation works and is able to replicate the results of one dataset of [1–3]. This aim was reached, indicating **Functional Suitability** of our implementation and also the **Reproducibility** of the original algorithm.[14]

[13] For better comprehension it is referred a StackOverflow question of the author of this thesis at `https://stackoverflow.com/a/69407977/5122790`

[14] If our implementation itself is reproducible can hardly be shown by the author of this thesis in this thesis.

Another goal was that the code-base successfully runs on the IKW compute grid. This goal was also met, and the resulting implementation and its **Automation** and **Scalability** proves satisfactory. Testing new hyperparameters is as easy as changing a YAML-file, transferring the file to the grid and executing a command[15]. The maximum number of cores per node and of nodes available to a user (64) are maximally utilized, which also indicates optimal dependency resolution. Running the algorithm on sample datasets worked without any complications in a matter of minutes (**Modularity, Maintainability, Adaptability**).

Allowing for all this was surprisingly intricate. Having worked with *Snakemake* on clusters before, the amount of customization to allow for workflow management on the IKW grid was surprisingly high. This was partially due to heavy iterative restructuring of the code-base to comply with the logic as demanded by this workflow management system. Part of this is due to peculiarities of its configuration,[16] but also to a huge degree due to the walltime-limit of 90 minutes. Because of this, most of the algorithm components need to be written in a way such that they both massiveley parallelize, but also gracefully end and store interim results, and the scheduler must comply with this. Comparing this with the uploaded implementation of [3][17], which consists of one Jupyter Notebook and one Python file, gives indication of the amount of work necessary to reach this. We sincereley hope that this thesis helps in **Transparency** and that at least the cluster execution will be re-used by other students of the IKW.

All the analyses that were conducted for this thesis are given in the accompaning source code. All plots and tables from all previous sections that were not reprinted were created in publicly available Jupyter-Notebooks, together with many more analyses that were conducted but would go far beyond the scope of this thesis. All plots and tables are explicitly linked and easily re-creatable and runnable. A lot of work was put into the architecture, and as soon as architechture and worklow management worked as intended, further development on the algorithm was suprisingly quick. This indicates that the general aim of creating an architecture that helps to answer related future research questions is fulfilled. For the sake of brevity, this analysis and the code-base itself, which is available at `https://github.com/cstenkamp/derive_conceptualspaces` shall suffice as explanation regarding the final two set goals.

5.4. Future Work

While this work has shown that the general technique seems to work, so far it has not produced an actual recommender for edcucational resources. Accordingly, building an interface as e. g. the one from Figure 1.1, or alternatively a textual interface that relies on a dialogue with a user to generate recommendations is an important direction for future work. When this is added to the Siddata-DSA, this may directly be combined with a study in which the users volunteer information about the usefulness of recommendation

[15] such as `MA_ENV_FILE=siddata.env submit by_config --configfile config/CONFIGFILE.yml`

[16] For example that *accounting files* that keep track of jobs are inaccessible to users, which means that our scheduler needs to simulate their behaviour.

[17] `https://github.com/rana-alshaikh/Hierarchical_Linear_Disentanglement`

or even providing possible labels, which may be used for algorithm evalution with labels other then the faculty, or even supervised training as done e. g. by [16]. Also, more analyses can be performed on the current data, such as correlating distances in the semantic space with *Levensthein-distances* in textual descriptions, or looking *in what direction* individual faculties differ.

Algorithm Addendums

As stated many times by now, the algorithm that was replicated in this work is very modular and requires only certain types of algorithm for many of its components. While recreating the works of [1–3], many smaller and larger changes to the algorithm could have been performed. A complete list of considered algorithm-extensions can be found in this repository[18]. Among others, not all extensions from [2, 3] have been implemented yet. It may also be insteresting to consider new distance-measures for the entities such as variations of the *Levensthein-distance* or the *Jaccard-distance* (e. g. IoU, as done in [25]). Also, more work is needed to remove irrelevant dimensions, in the hope of making the algorithm more robust. Especially the clustering and merging of similar features may benefit from considering other techniques. These include using IoU similar to [35] as similarity measure, methods to remove uninformative clusters, or considering other algorithms for clustering and calculating the centroid directions that weight or threshold based on classifier performance or similarity. Finally, finding a representative cluster name may benefit in robustness from also considering pseudo-documents [16] or other methods, e. g. those suggested by Carmel *et al.* [54].

Complex Changes

Besides the aforementioned additions, there are also many avenues that change more of the algorithm's logic. These include the fine-tuning step of Ager *et al.* [2], which may be combined with latent topic detection techniques to alleviate the fact that the desriptions of the Siddata-dataset are short and omit many words that may serve to describe an entity. Many other ways to find latent topics may also be considered, such as relying on semantic databases and counting *hyponyms* of candidate directions towards their count for a description, or again relying on vector similarity as demonstrated in LSA and also neural embeddings such as Word2Vec and Doc2Vec. We have established that the current form of the algorithm only yields either a space with a Euclidean metric or one with semantic directions. Finding ways to get both would be an important contribution - it is e. g. unclear why so far only rankings have been considered instead of linear algebra to find a new coordinate base. Alternatively, one can ignore the requirement of the Euclidean metric completely, which opens up many major changes to the algorithm such as not relying on BoW-representations at all, but instead using modern document embeddings such as BERT, where candidates are detected using the cosine distance.

[18] `https://github.com/cstenkamp/MastersThesisText/blob/master/pandoc_markdown_base/futurework_long.md`

New Algorithm

Finally, let us try to combine all the issues raised in this work to form a suggestion of how a completely different algorithm may look. First, we want an embedding that retains as much of the original dataset variability as possible. For that, additionally to a small set of absolutely most important dimensions found as the orthogonal of the linear classifiers, one may consider subsequently running PCA[19] to find *remaining* variance for those dimensions that do not obviously correspond to the occurance of single words. This has the additional benefit of finding *orthogonal* directions which helps to get low-dimensional spaces, expressed by a small amount of uncorrelated features. So far, the directions found by PCA do not have a name, but we have also examined ways such as LSA that find words by their similarity of angles. In fact, it may be possible to build the whole algorithm on the basis of PCA: Instead of relying on BoW-representations, texts could be embedded using e. g. BERT, which also takes into account its hidden topics and word ordering. Subsequently, one may run PCA to find a few directions of this space that best explain the dataset variance. Embracing the fact that one is dealing with vectors, LSA can then be used to find candidate dimension names which do not rely on exact phrasing of the corpus. A disadvantage of such an algorithm is, that the resulting space is not metric[20] - but neither is the space yielded by the algorithm explored in this thesis.

5.5. Conclusion

This thesis explored how explainable generation of educational resources can be generated in a data-driven manner from a corpus of their descriptions. It has been established that *Conceptual Spaces* can serve as knowledge representation method for that by capturing dataset properties in a way that allows to computationally model explainable reasoning. To generate these automatically, the method by [1] was thoroughly examined in theory and in practice.

The introduced algorithm was successfully replicated and the results on the originally used data have been confirmed. We have sucessfully transferred the algorithm to the domain of educational resources and demonstrated its ability to find relevant human concepts in our dataset. By means of this, we have demonstrated the robustness of the methodology by discussing the difference in the datasets as well as the respective results. Some adjustments to the original algorithm have been shown to further increase its performance. Analysis on the results of our dataset has shown that surrogate metrics indicate that the algorithm finds some regularities in the dataset, such as a courses' faculty. Despite this, more work to thoroughly test the algorithm's capabilities, but also to increase its performance and robustness will be needed.

To make the replication and application of the algorithm possible, we have implemented a tool that allows to easily recreate the results of [1–3]. This tool was built while ensuring to comply with code quality standards and proper methodology, is open-sourced and

[19] PCA is a technique used for dimensionality reduction that works by projecting high-dimensional data onto its *Principal Components*, those directions that explain most of its variance.

[20] Though one may try to use pairwise distances for each of the dimensions to create a new metric space using a variation of MDS, or rely on Kohonen-Networks for that, as suggested by Gärdenfors [18].

can easily be installed as a package. The result of this is a reliable pipeline that has been demonstated to be easily adaptapted and extended. We hope this helps raising the attention of a broader community to the methodology and enables future research on further varition or domain transfers to be simple, quick and accessible. Another major contribution of this thesis are the solutions that were found to run workflows similar to the one required here on compute clusters, specifically the IKW grid. The result is a workflow that is much more versatile and user-friendly than the ones we know to be used to date,[21] and we hope that it helps to make high performance computing more attractive to other UOS students. All analyses conducted in this work are open-sourced as well and can be easily inspected and re-created.

Finally, we have discussed Conceptal Spaces and Derrac & Schockaert's [1] algorithm extensively on the basis of our results. We have reached the conclusion that while the algorithm produces reasonable results automatically and shows much potential, it drops necessary assumptions of Conceptual Spaces by not producing a single embedding that is both metric and also has interpretable dimensions. We have concluded that because of this, some assumptions about necessary design decisions can be dropped, which may allow to combine the methodology with state-of-the-art algorithms.

[21] So far, the only methods that we have seen suggested are to write custom shell-scripts, see e. g. the discussion and links in its respect Stud.IP group at `https://studip.uni-osnabrueck.de/plugins.php/coreforum/index?cid=e946790f6a74e17df02a85847b2110ab` (accessed at 9th April 2022)

Glossary

Definitions

Notation	Description	Page List
accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	56
cosine distance	$\cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$	3, 21, 25, 42, 45, 83, 85, 86, 88
dissimilarity matrix	A square matrix where both rows and columns represent entities, the cells being to their pairwise dissimilarities as calculated by an arbitrary distance function. For metric distances, distance matrices are mirrored along their main diagonal, which is made up solely from zeros. Also called **distance matrix**.	25, 36, 42
Doc2Vec	doc2vec (or *Paragraph Vectors*) refers to a technique by [14] that represents a document by a dense vector that is trained to predict the word occuring in it, analaogous to the training of word2vec. In contrast to bag-of-words-representations of texts, it considers word order and semantics of the words, which often leads to substantial improvements e. g. in classification and information retrieval tasks. Nowadays, there are many better performance models based on *transformer* ANN architectures such as BERT [15] wich base on similar training techniques but outperform in such tasks.	35, 42, 83, 88
document-term matrix	A document-term matrix encodes the frequency of terms (words, n-grams or other) for a collection of texts in a matrix. The (often very sparse) matrix has a rows represending the documents and columns corresponding to terms, the individual values encoding the pure counts, frequencies or quantifications of all combinations of document and term.	22, 23, 40, 41
F-1 score	$2 * \frac{precision*recall}{percision+recall} = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$	56
lemma	The **lemma** of a word is the canonical, base form of a set of words belonging to the same lexeme. **Lemmatizing** a word refers to the process of finding this base form for (possibly inflected) words. For example, the lemma of the words *going, went, gone* is *go*.	21, 39

Notation	Description	Page List
n-gram	n-grams are sequences of consecutive words of length n. For example, the text "I eat lunch" contains the 1-grams ["I", "eat", "lunch"], the 2-grams ["I eat", "eat lunch"] and the 3-gram ["I eat lunch"]. In the scope of this thesis, the term **phrase** also refers to n-grams.	21, 22, 34, 36, 39, 41, 44, 73, 102
stop word	**Stop words** are words that are very common to a language and thus of low discriminative power for differentiating between individual texts of a corpus. Stop word lists are used as negative dictionary to remove them before text processing. There are no universally agreed-upon stop word lists.	38, 103
t-SNE	t-SNE is a dimensionality-reduction algorithm often used to visualise high-dimensional data in two or three dimensions. It converts distances between data point to probabilities and minimizes the pairwise Kullback-Leibler-divergence between the joint probabilities of the original data and their respective embeddings.	, 34, 56, 60, 74, 106, 107
Word2Vec	word2vec is the most famous of a family of *neural language models* [38]. These models are trained on large corpora of texts to predict a word from its surrounding words or vice versa. Each word is represented as a vector, and the training ensures that semantically similar words end up with similar vector representations. This helps with many NLP tasks, as it counters problems of synonymy and polysemy by considering context. The difference between vectors carry semantic meaning, however unlike conceptual spaces they are not domain-spefic and embed all natural language words in a high-dimensional space of arbitrary dimensions.	4, 21, 47, 83, 86, 88

Custom Terms

Unlike the *Definitions*, this section of the glossary contains terms that used in this thesis that will have the same meaning throughout the thesis, but are no official definitions of the term outside of this work.

Notation	Description	Page List
entity	An entity is a single sample from the handled corpus. Depending on the context, this term may also refer to its associated text (which may, depending on the considered dataset, be the course-description, picture-tags, concatenated-reviews, ...).	14, 27, 34, 35, 39, 41, 46, 56, 85, 93
feature-based representation	In the context of the given algorithm, the **feature-based representation** of an entity is its representation as a feature-vector, where each dimension corresponds to a semantic feature and its value is the entity's respective rank for that feature.	54, 93
hyperparameter	When it is referred to *hyperparameter* in this work, it does not only refer to scalars like the dimensionality of an embedding, but also e. g. which specific algorithm is used in a step of the algorithm.	48, 50, 52
quantificiation	In the scope of this thesis, the term **quantification** refers to the relative score for an n-gram in a document, depending on its frequency as well as other frequencies, as calculated by one of the 2. Word-weighting techniques, also called **quantification measures**.	20, 22, 40, 42, 44, 46, 65, 72, 80
rank	A ranking of a set up numbers refers to their respective index when ordered-by-value. In this work it refers specifically to the value of an entity with respect to a semantic direction. Relevant in the feature-based representation.	41, 47, 54, 61, 93

Acronyms

The abbreviations used throughout the work are compiled in the following list below. Note that the abbreviations denote the singular form of the abbreviated words. Whenever the plural forms is needed, an *s* is added. Thus, for example, whereas ANN abbreviates *artificial neural network*, the abbreviation of *artificial neural networks* is written as ANNs.

Notation	Description	Page List
ANN	Artificial Neural Network	11
BERT	Bidirectional Encoder Representations from Transformers	19
BoW	Bag of Words	14
CL	Computational Linguistics	
CLI	Command Line Interface	49

Notation	Description	Page List
CS	Conceptual Space	11
DAG	Directed Acyclic Graph	49
DDC	Dewey Decimal Classification	19
df	Document Frequency	27
DSA	Digital Study Assistant	10
DT	Decision Tree	
IKW	Institut für Kognitionswissenschaften	6
IMDB	Internet Movie Database	14
IoU	Intercept over Union	88
LDA	Latent Dirichlet Allocation (see Subsection 2.4)	57,
LMS	Learning Management System	10
LSA	Latent Semantic Analysis	23
LSI	Latent Semantic Indexing	23
MDS	Multi Dimensional Scaling	14
ML	Machine Learning	7
MOOC	Massive Open Online Course	2
NLP	Natural Language Processing	4
OER	Open Educational Resource	2
OS	Operating System	48,
PCA	Principal Component Analysis	85
POS	Part-of-Speech	40
PPMI	Positive Pointwise Mutual Information	14
RCC	Region Connection Calculus	16
SGE	Sun Grid Engine	49
SVM	Support Vector Machine	15
t-SNE	t-distributed Stochastic Neighbor Embedding	
tf	Term Frequency	
tf-idf	term frequency × inverse document frequency algorithm	14
UOS	University of Osnabrück	30
VSM	Vector Space Model	4

Appendix

A. Code Use-Cases in Praxis

This appendix lists complete an working commands on how exactly to invoke the pipeline described in this thesis. There are three main ways to invoke this codebase: Running the complete pipeline using **Snakemake**, running individual components for development or debugging using the CLI of **Click**, or inspecting results in **Jupyter-Notebooks**. The following sections list some examples for all of these use-cases.

A.1. Docker

As the code is both compiled as a python-package as well as a docker-container, it is possible to invoke all commands listed here for those options as well. For information on how to install a docker-container it is referred to `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/doc/install_docker.md`. To build the docker-container, use the following command:

```
docker build -f $MA_CODE_BASE/Dockerfile --build-arg git_commit=$
    (git rev-parse --short HEAD) --build-arg uid=$(id -u) --build-
    arg gid=$(id -g) --rm --tag derive_conceptualspaces
    $MA_CODE_BASE
```

Afterwards, you can make an alias for the codebase inside a container:

```
alias run_ma="docker run -it --rm --user $(id -u):$(id -g) --name
     derive_conceptualspaces_cont -v $MA_DATA_DIR:/opt/data --env-
    file $MA_ENV_FILE derive_conceptualspaces"
```

And then call anything from inside the container with e. g.

```
MA_SNAKEMAKE_TELEGRAM=1 ma_cont snakemake --cores 3 -p  --
    directory /opt/data default
ma_cont python -m derive_conceptualspace generate-conceptualspace
     create-candidate-svm
```

A.2. Using Click

The `click` python package is used to generate a command-line interface. The structure of a command is recursive: `command \textrightarrow subcommand \textrightarrow subsubcommand`. This perfectly mirrors the structure of this codebase, as every subcommand can accept more configurations and accept another nested set of dependencies (interim results or results from earlier steps) as argument. At any level of subcommand you can run `--help` to get a list of available arguments and commands.

Minimal way of calling it: `python -m derive_conceptualspace generate-conceptualspace create-candidate-svm` if you have installed the package.

A.2.1. Passing configurations

The algorithm allows for many different parameters or selection of sub-components. `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/derive_conceptualspace/settings.py` contains the default values for the most important parameters, which can be also set in many different ways. Next to the configurations in the `settings.py` file, there are also other configurations that can set when executing this via click, such as `--log` selecting python's log-level. Again, it is encouraged to run any command with `--help` to figure out all possible arguments.

This code-base allows several ways to inform about demanded configurations, which are listed below. Note that this list is ordered by priority of the arguments, so you can combine all the ways below, where the upper ways to set arguments overwrite the lower ones, but you will be warned if a setting is overwritten by a higher higher priority.

- With command-line-arguments. Every command specifies a list of allowed command-line-arguments, such as eg. the command `preprocess-descriptions` allowing the `--language` argument.
- With `--env-file`: Either by giving a command-line-arg that refers to a `.env`-file, or by having the `MA_ENV_FILE` environment-variable set when executing the code. The latter way is the fastest way to change settings for all sub-commands in the development-process: An additional `select_env.env` file can refer to the actual env-file. This ensures that the run configurations for all commands are changed simultaneously and also allows to quickly select a new suite of arguments for individual datasets - which automatically selects new parameters for all sub-commands. See below for the schema of these environment variables.
- With `--conf-file`, referring to a `YAML`-file with configurations. Note that this argument can, again, be an environment variable. Note that when providing several alternatives for a configuration, this mode will select one consistent sample instead of spawning a process for each possible value, which differs from the behaviour when calling it with Snakemake. `https://github.com/cstenkamp/derive_conceptualspaces/tree/main/config` contains a number of files with parameter-combinations corresponding to the exact configuration used by any of [1–3].
- With correct env-vars already set: When not overwritten by one of the above ways, configurations are automatically drawn from environment-variables if the respective variable exists in the current context. Note that when using environment-varibles, they must be prepended with `MA_`: To overwrite the configuration `PRIM_LAMBDA`, you would need to set the variable `MA_PRIM_LAMBDA`.
- From default-values. If no of the above ways was used to overwrite a configuration, a default-value is used as specified in the `settings.py`.

If a given command relies on interim results, it must be figured out which of the candidates is selected. To resolve the dependencies, the configurations are resolved as described above, and the dependency that exactly fulfills the required configuration is selected. If the dependency itself relied on more configurations, the values for these are added to the state with maximal priority. If their value is to be overwritten lateron, the codebase figures out if there are any conflicts and either gracefully fails or informs of their difference.

The latter may be relevant when for example only requiring `DEBUG=True` for a later step of the algorithm.

A.2.2. Sample usage

With this as background, let us give some samples of how the code is invoked. First, let us look at the respectively used configurations:

Used Configurations

Contents of `/path/to/select_env.env`**:**

```
MA_BASE_DIR=$HOME/path/to/data
MA_CODEPATH=$HOME/path/to/code
MA_CONDAPATH=$HOME/miniconda3
MA_CONDA_ENVNAME=create_cs
MA_CUSTOM_ACCTFILE=$HOME/custom_acctfile.yml
MA_CONFIGDIR={MA_CODEPATH}/config
MA_GRIDCONF={MA_CODEPATH}/workflow/sge/ikw_grid/
MA_ENV_FILE=siddata.env
```

Note that this file contains mostly paths. Separating these from the actual configuration is useful especially when running the code on different machines, such as a local machine used for testing plus a high-performance grid or cluster to be deployed to. Using `MA_ENV_FILE`, it is referred to the actual environment-file. This can contain settings specific to individual datasets, such that only this line must be exchanged to inform all possible commands simultaneously of the configurations required for other datasets.

Contents of `/path/to/code/config/siddata.env`**:**

```
MA_DEBUG=1
MA_DEBUG_N_ITEMS=500
MA_DATASET=siddata2022
MA_MIN_WORDS_PER_DESC=80
MA_LANGUAGE=de
MA_CONF_FILE=derrac2015.yml
```

This file is used to set configurations that are relevant only to one specific dataset. Furthermore it links a `MA_CONF_FILE`, that is used to set configurations that are *global* in the sense that they affect the configuration irrespective of the used dataset. The contents of `/path/to/code/config/derrac2015.yml` that allow to recreate the configuration of [1] is listed in Section B.3.1.

Sample Inputs and Outputs

The following examples explicitly export the `MA_SELECT_ENV_FILE` environment variable, but keep in mind that the `python_dotenv` package or you IDE's Plugin to consider environment-files are just as valid. The PyCharm-Run-Configurations used in the development can be found at `https://github.com/cstenkamp/derive_conceptualspaces/tree/main/.run`

Create dissimilarity-matrix without command-line-arguments:

```
export $(cat $MA_SELECT_ENV_FILE | xargs)
python -m derive_conceptualspace generate-conceptualspace create-
    spaces create-dissim-mat
>> Starting up at 23.03.2022, 16:13:52
>> Config-File .../Derive_Conceptualspace/config/
    derrac2015_edited.yml loaded.
>> Running with the following settings [3ef0e2c137]:
    CANDIDATE_MIN_TERM_COUNT: 2, CLASSIFIER: SVM,
    CLASSIFIER_SUCCMETRIC: kappa_rank2rank_onlypos_min, DATASET:
    siddata2022, [...]
```

Extract candidate-terms, enforcing the usage of KeyBERT:

```
export $(cat $MA_SELECT_ENV_FILE | xargs)
export MA_PP_COMPONENTS=mfauhtcsldp;MA_TRANSLATE_POLICY=onlyorig;
    MA_DEBUG=1
python -m derive_conceptualspace prepare-candidateterms --
    extraction-method keybert extract-candidateterms --no-faster-
    keybert
>> Starting up at 23.03.2022, 16:35:16
>> Config-File .../Derive_Conceptualspace/config/
    derrac2015_edited.yml loaded.
>> Debug is active! #Items for Debug: 500
>> Using a random seed: 1
>> conf_file demanded config EXTRACTION_METHOD to be tfidf, but
    cmd_args overwrites it to keybert
>> The setting DEBUG was False in a dependency and is True here!
>> Running with the following settings [82fe5f58dd]:
    CANDIDATE_MIN_TERM_COUNT: 2, CLASSIFIER: SVM, [...]
```

A.3. Using Snakemake

This code-base uses Snakemake to allow running the entire pipeline at once. This can be run directly on a local machine, but also scheduled on e. g. the Sun Grid Engine. For that, each base command is specified with some required metadata in the Snakefile at `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/workflow/Snakefile`. Snakemake can be invoked with many arguments to for example specify the maximal number of cores to consider, specifying the used conda-environment using `--use-conda`, or to make Snakemake continue running if a single rule failed using `--keep-going`. For all possible arguments, it is referred to `https://snakemake.readthedocs.io/en/stable/executing/cli.html#all-options`. Besides these arguments, this codebase provides a large set of to invoke and configure the pipeline.

If this codebase is installed as a package (using `pip install`), snakemake can be called exactly as described above. Otherwise, the codebase may not be in your operating system's `PATH`, which means that you may need to set the env-var `PYTHONPATH=$(realpath /path/to/code):$PYTHONPATH`. It is also possible to run snakemake from a container, using a command similar to the one specified above.

A.3.1. Snakemake Use Cases

There are the following ways to invoke the pipeline with Snakemake. Note that to consider an env-file, you can simply export their values first first:
`(export $(cat $MA_SELECT_ENV_FILE | xargs); snakemake --cores 1 -p default)`.

default e. g. `snakemake --cores 1 -p default`
Runs a single configuration, namely the default configuration as specified by default-values or currently active environment variables.

all e. g. `snakemake --cores 1 -p all --keep-going`
Runs all configurations from the cartisian product of all allowed values for all configs. Which values are allowed can be specified in the `settings.py` by replacing the definition `DEFAULT_VARNAME` by a collection `ALL_VARNAME`, as can be inspected here: `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/derive_conceptualspace/settings.py#L32-L40`. Note that this is a huge combinatorical explosion and only recommended with `DEBUG=True`!

all_for e. g. `snakemake --cores 1 -p all_for --config for_rule=create_embedding`
Runs the cartesian product of all configuration similarly to **all**, however only executes the pipeline up to a specific step, in this example the rule `create_embedding`.

by_config e. g. `snakemake --cores 1 -p by_config --configfile ./config/derrac2015.yml --keep-going`
Is used to run all configurations specified in a config-file. If any of the values of this configuration-files is a list, all these values will be run individually, allowing to execute the pipeline for different parameter-combinations simultaneously (note the different behaviour when using `click`). This configuration-file further allows to specify configurations specific to a dataset, as shown in Section B.3.1.

specific files e. g. `snakemake --cores 1 -p --directory $MA_DATA_DIR siddata/debug_True/fautcsdp_translate_minwords100/embedding_ppmi/dissim_mat.json`
The original way how Snakemake-workflows are invoked. This allows to specify only those configurations that are part of the file-path, resolving configurations and dependencies by the expected positions in it.

A.3.2. On Grids/HPCs

Additionally to running on a single machine, it is also possible to submit this workflow to a cluster or grid. There is a lot of code to allow to either *schedule* this workflow on the IKW grid, or to just run it manually. To run it manually, you can use the `run_manually` script, by e. g. `/path/to/code/workflow/sge/run_manually.sh by_config --configfile /path/to/code/config/derrac2015.yml /path/to/code/workflow/sge`.

To schedule the workflow, you first have to install the environment. This can be done by `qsub`ing the `workflow/sge/install_env.sge`-file. Afterwards you may execute `MA_SELECT_ENV_FILE=/path/to/code/config/select_env.env /path/to/code/workflow/sge/submit.sh`. The submit-script takes care of forwarding all envirionment variables and parsing the actual `ENV_FILE` as specified in the `SELECT_ENV_FILE` (even allowing to nest env-vars such as `MA_CONFIGDIR={MA_CODEPATH}/config`). It further provides arguments to

kill all already running instances (`-k`), watch the progress after scheduling it (`-w`), and to remove all old logs and outputs (`-r`). A full sample call is for example `MA_ENV_FILE =placetypes.env submit -kwr by_config --configfile config/derrac2015.yml`. Internally, this script calls `snakemake -p by_config --configfile <configfile> --directory $DATAPATH --profile /path/to/workflow/ikw_grid/sge`. Configurations specific to the grid engine are, next to the `Snakefile`, specified in a `cluster.yaml` file (`https://github.com/cstenkamp/derive_conceptualspaces/blob/main/workflow/sge/ikw_grid/cluster.yaml#L13-L20`). This allows to specify the default amount of memory, and also if there is a wall-time such as for the IKW-Grid. This runtime is considered by restarting both the scheduler, and also passing it to the individual runners for them to gracefully shutdown and restart.

More information on scheduling Snakemake is available in the respective repository (`https://github.com/cstenkamp/Snakemake-IKW-SGE-Profile`) and the How-To (`https://github.com/cstenkamp/derive_conceptualspaces/blob/main/workflow/sge/howto.md`)

A.4. In Notebooks

Inspecting code in Notebooks requires a different handling of contexts, as there is not one single context to which new elements are added. Instead, many different contexts are supposed to be loaded quickly and easily for inspection. This code base provides many helper-functions to get individual configurations, the **best** configuration according to a specifiable metric, or to load many individual configurations at once.

Additionally, many helper functions improve the ease of use for loading configurations from `.env`-files and configuration-YAMLS, help in recovering the environment-variables necessary to load a configuration using click, and much more. A sample call to load all configurations for a dataset looks like this:

```
setup_logging()
load_envfiles("siddata")
configs, print_cnf = getfiles_allconfigs("clusters", verbose=True
    )
>> There are 165 different parameter-combis for dataset
    siddata2022:
>> {'dataset': 'siddata2022',
>> 'language': 'de',
>> 'debug': 'False',
>> 'pp_components': ['mfauhcsd2', 'mfauhtcsldp'],
>> [...]
print_envvars(get_filename(configs[0], get_dependencies=False))
>> MA_DATASET=siddata2022;MA_LANGUAGE=de;MA_DEBUG=False;
    MA_PP_COMPONENTS=mfauhcsd2 [...]
```

To for example get the interim result `featureaxes` of all configurations, you may use this:

```
with WorkerPool(DEFAULT_N_CPUS-3, pgbar="Fetching featureaxes..")
    as pool:
    get_featureaxes = lambda conf: ((ctx := SnakeContext.
       loader_context(config=conf, silent=True)).
       get_important_settings(), ctx.load("featureaxes"))
    featureaxes_list, interrupted = pool.work(configs,
       get_featureaxes)
```

The final sample shows how to find and inspect a best-performing parameter-combination according to a decision-tree, and then running functions on interim results as well as inspecting old outputs:

```
setup_logging()
load_envfiles("placetypes")
conf, perf = get_best_conf("Geonames", verbose=True,
   balance_classes=True, one_vs_rest=True, dt_depth=1,
   test_percentage_crossval=0.3, metric="f1")
ctx = SnakeContext.loader_context(config=conf, silent=False)
pp_descriptions, filtered_dcm = ctx.load("pp_descriptions", "
   filtered_dcm" loaders=dict(pp_descriptions=DescriptionList.
   from_json)
ctx.obj["filtered_dcm"].show_info(descriptions=ctx1.obj["
   pp_descriptions"])
>> [...]
ctx.display_output("pp_descriptions")
>> [...]
```

B. Implementation Details

A main goal of this thesis is to provide a code base that makes it as simple as possible to get started with Derrac & Schockaert's [1] algorithm to derive rudimentary conceptual spaces for any kind of dataset. In order to achieve this, documenting some implementation details and design decisions is crucial. This appendix goes into more detail for selected components of the algorithm.

B.1. Algorithm Implementation Details

Preprocessing

Language-Detection and Translation

To check the languages of the entities, the `langdetect` [1] library is used, which is a direct port of a java library that claims to have 99.8% accuracy on longer texts [58].

Depending on the translation-policy, it is possible to either take only those entities of the demanded language, ignore it and consider all entities in their original language, or enforce the demanded language by translating all entities from their original language to the demanded one. The accompaning code for this thesis contains extensive code to do that using the *Google Cloud Translation API* [2]. Many descriptions of the Siddata-dataset were translated using this technique[3]. As of now, Google's Cloud Translation Service uses an embedding-based neural model of a hybrid architecture that has a transformer encoder, followed by an RNN decoder [44]. All of the languages detected in the Siddata-dataset are supported by the system - translating between the languages German, English and Spanish, which make up more than 99 percent of the Siddata-descriptions, is what the system is particularly optimized for.

Candidate Extraction

KeyBERT

The *KeyBERT*-algorithm[4] [53] is one of the techniques used to select phrases of the text-corpus as candidates for feature-directions.

KeyBERT is a keyword-extraction technique "that leverages BERT embeddings to create keywords and keyphrases that are most similar to a document"[4]. BERT is a neural language representation model that is able to embed both words and documents. Its embeddings are obtained by training a multi-layer bidirectional transformer encoder ANN architecture on a task in which a masked word must be predicted from the its bidirectional context as well subsequent fine-tuning tasks [15]. To extract keywords, the KeyBERT algorithm embeds both the document as well as its containing n-grams of a configurable

[1] `https://pypi.org/project/langdetect/`, Shuyo [58]

[2] `https://cloud.google.com/translate`

[3] As, however, only 500.000 characters per google-account and month can be translated free of charge, the translation-process for the descriptions is still in progress.

[4] Grootendorst, M. *KeyBERT* `https://github.com/MaartenGr/KeyBERT`. 2021

length using BERT and returns those phrases whose embedding ist most similar to the document-embedding according to the cosine-similiarity[4].

The KeyBERT-model was incorporated to extract key-phrases for this codebase in two ways:

KeyBERT-original runs the algorithm on the unprocessed original texts. This is reasonable, as this is what BERT-embeddings are trained on, however it has the disadvantage that it requires a lot of post-processing to match the extracted phrases to the processed descriptions (which e. g. may contain only lemmas or have their stop words removed)

KeyBERT-preprocessed alleviates this problems by running the algorithm on already preprocessed texts. This may however lead to worse results, as the algorithm was trained on unprocessed natural sentences.

In practice, though both variants extracted different phrases, the results for either of the technqiues did not differ significantly.

Candidate Filtering

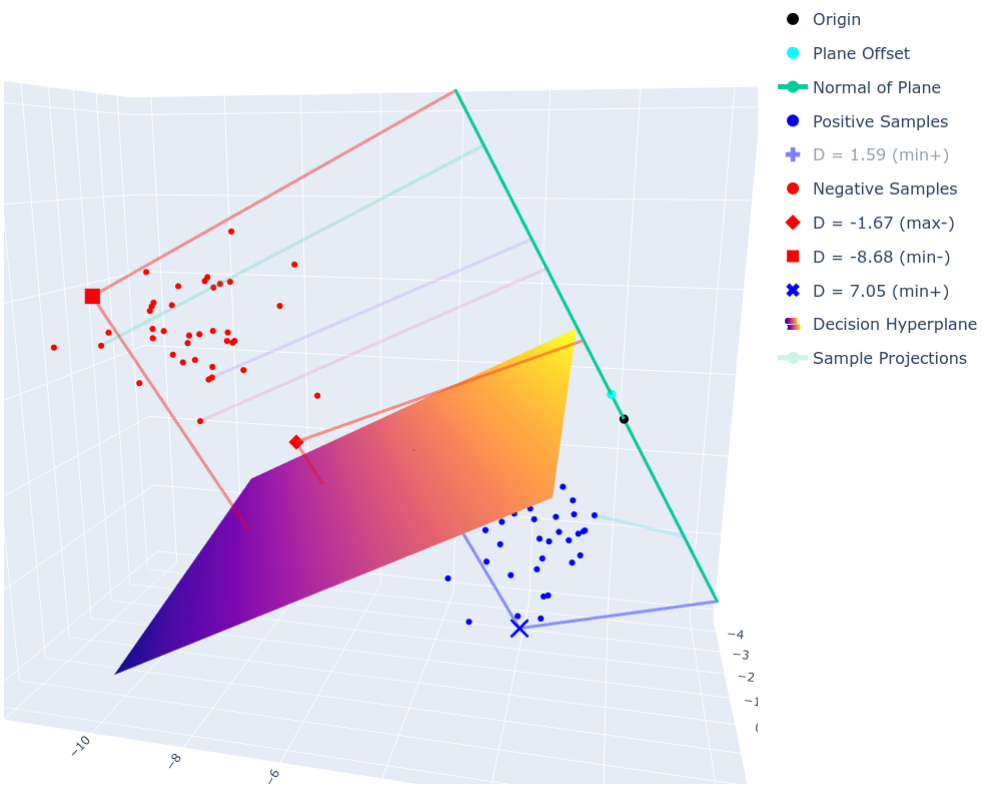

Figure B.1.: Visual representation of the Hyperplane of a Support-Vector-Machine splitting a dataset, as well as it's orthogonal and the orthogonal projection of a set of samples onto the plane. For an interactive version of this plot, visit `https://nbviewer.org/github/cstenkamp/derive_conceptualspaces/blob/main/notebooks/text_referenced_plots/hyperplane_orthogonal_3d.ipynb?flush_cache`

Long	Short	Data	Quantifications	Distances	Comments
rank2rank_dense	r2r-d	all	Dense-Ranked	Dense-Ranked	
rank2rank_min	r2r-min	all	Min-Ranked	Dense-Ranked	
bin2bin	b2b	all	Binary	Binary	Disregards rankings
digitized	dig	all	Digitized	Digitized	10 bins. Positions decided by np.histogram_bin_edges from min and max of all data
count2rank_onlypos	c2r+	positive	Unchanged	Dense-Ranked	Only for Count as Quantification
rank2rank_onlypos_dense	r2r+d	positive	Dense-Ranked	Dense-Ranked	
rank2rank_onlypos_min	r2r+min	positive	Min-Ranked	Min-Ranked	
rank2rank_onlypos_max	r2r+max	positive	Max-Ranked	Max-Ranked	
digitized_onlypos_1	dig+1	positive	Digitized	Digitized	10 bins. Positions by np.histogram_bin_edges from min and max of all data
digitized_onlypos_2	dig+2	positive	Digitized	Digitized	10 bins. Positions by np.histogram_bin_edges from min and max of all positive data

Table B.1.: Different values of calculating Cohen's Kappa score. Dense means: if there are 14.900 zeros, the next is a 1. Min means: if there are 14.900 zeros, the next one is a 14.901. Max means: if there are 14.900 zeros, they all get the label 14.900. These scores are weighted.

B.2. Other Algorithms

B.2.1. Semantic Knowledge Bases

Lexical databases of semantic relations between words, the most famous of which being WordNet,[5] link words in a graph that encodes explicit semantic relations like synonyms and hyponyms (subtypes/ *is-a*-relationships). While neural embeddings may encode similar information implicitly, when relying on dictionary-based word encodings they are an important tool when using classical linguistic techniques. For the developed algorithm, the information how many hyponyms of a candidate word for a semantic direction occur in its corresponding text-corpus can be highly relevant. To do that, WordNet [60] and it's German equivalent, GermaNet [61, 62],[6] are used in the respective step.

B.2.2. Faculty-Classifier

As one of the evaluations is to compare the results of classifiers based on the algorithm here with a powerful classification algorithm, a neural network that classifies the faculty of a course in the Siddata-Dataset was also implemented. The implementation for that will not be elaborated upon except that it uses Google's 'universal-sentence-encoder-multilingual' in Version 3 (linear in textlength, thus managable time and space requirements) plus a few classification layers ontop. The encoder is trained "on a number of natural language prediction tasks that require modeling the meaning of word sequences rather than just individual words",[7] aimed being the base for architectures for many NLP tasks through the usage of sentence embeddings [43]. It was trained on with a train-test-split of 90/10 (the results being consitent through sampled cross-validation)

Another purpose of the classifier is to check if it is anyhow possible to extract meaningful information from the descriptions: If it is possible to train a classifier on the data that

[5] `https://wordnet.princeton.edu/`

[6] `https://uni-tuebingen.de/fakultaeten/philosophische-fakultaet/fachbereiche/neuphilologie/seminar-fuer-sprachwissenschaft/arbeitsbereiche/allg-sprachwissenschaft-computerlinguistik/ressourcen/lexica/germanet-1/`

[7] Quote from their description at `https://tfhub.dev/google/collections/universal-sentence-encoder/1` (accessed at 25th March 2022)

can reasonably predict a qualitative feature, there is enough structure in the data such that the algorithm we are about to produce can work. Also, we have a lower bound for useful data: we can just throw away data that cannot be classified.

B.3. Configurations to run [1, 2]

B.3.1. Derrac & Schockaert [1]

Listing B.1: YAML for Derrac & Schockaert [1]

```
pp_components:              mfautcsdp
quantification_measure:     ppmi
dissim_measure:             norm_ang_dist
embed_algo:                 mds
embed_dimensions:           [20, 50, 100, 200]
extraction_method:          pp_keybert
max_ngram:                  5
dcm_quant_measure:          count
classifier:                 SVM
classifier_succmetric:      [kappa_count2rank_onlypos,
  kappa_rank2rank_onlypos_min]
prim_lambda:                0.5
sec_lambda:                 0.1
__perdataset__:
  placetypes:
    extraction_method:      all
    pp_components:          none
```

B.3.2. Ager *et al.* [2]

- Logistic regression instead of SVM
- Hyperparameters for metric: Kappa, Accuracy, NDCG
- Ranking is of the PPMI score in the BoW
- N best-scoring candidate features, with either Derrac's algorithm or k-means
- Centroid of the cluster = Average of the normalized vectors of the words
- Semantic spaces created by either of
  - MDS from angular differences of PPMI-weighted BoW
  - PCA from angular differences of PPMI-weighted BoW (no quadratic time)
  - Doc2Vec [14]
  - Averaged pretrained GLoVe of words with df ≥ 2
  - Averaged pretrained GLoVe, weighted by PPMI
- ndims of embedding one of 50, 100, 200
- Number of input-dimensions for clustering: 500, 1000, 2000
- Number of clusters {k, 2k} with k being the input-dimensions

C. Further Plots and Tables

C.1. t-SNE plots for the data from [1]

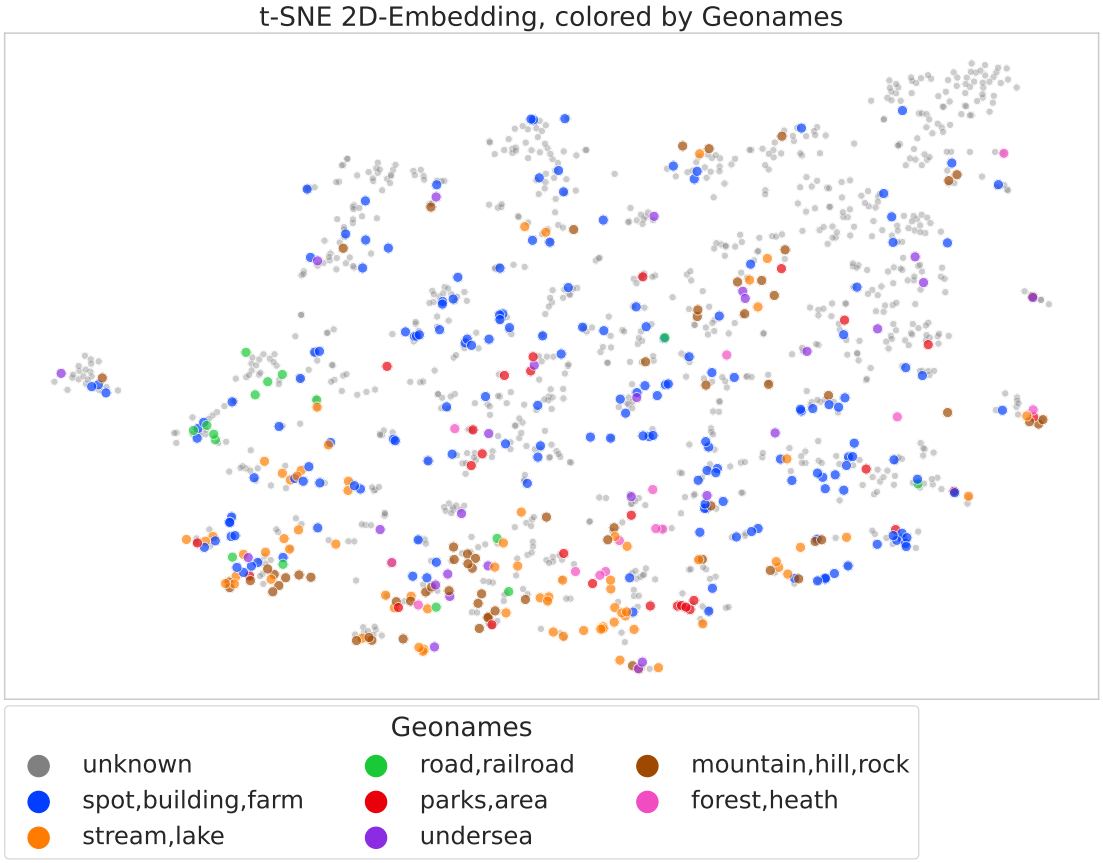

Figure C.1.: 2D visualization of the placetypes-dissimilarity-matrix for the data uploaded by Derrac & Schockaert [1], colored by GeoNames. Generated with t-SNE. See `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/notebooks/text_referenced_plots/desc15_mds_2d3d.ipynb` for the origin of this plot. Individual class frequencies: spot,building,farm: 176 | stream,lake: 74 | mountain,hill,rock: 68 | parks,area: 28 | undersea: 27 | road,railroad: 16 | forest,heath: 14

Note that at `https://github.com/cstenkamp/MAAnalysisNotebooks/blob/main/analyze_results/derrarc2015/mds_derrac2015_moviesplaces_2d3d.ipynb` there are also interactive 3D-Versions of these plots. The one for the Siddata-dataset is at `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/notebooks/text_referenced_plots/visualise_embeddings.ipynb`, also 2D and interactive 3D.

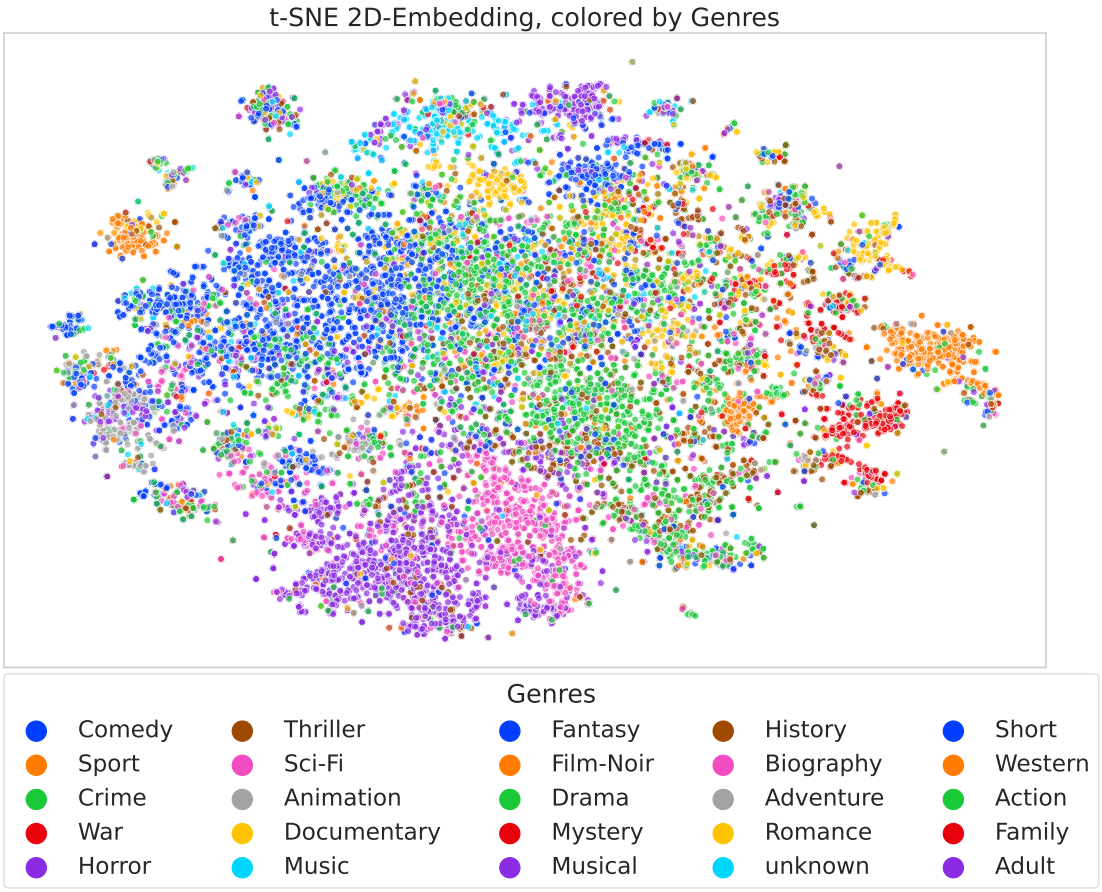

Figure C.2.: 2D visualization of the movies-dissimilarity-matrix for the data uploaded by Derrac & Schockaert [1], colored by genre. Generated with t-SNE. See `https://github.com/cstenkamp/MAAnalysisNotebooks/blob/main/analyze_results/derrarc2015/mds_derrac2015_moviesplaces_2d3d.ipynb` for the origin of this plot.

C.2. Dataset samples

Number	Course
1.003	Spez. Soz. III: Wahlforschung (Politische Soziologie)
1.003	Erziehung, Bildung und Sozialisation in der modernen Gesellschaft
1.003	Infoveranstaltung Master Soziologie
7.441202	"Fremdheit" in der Kinder- und Jugendliteratur
7.441202	„Mündlichkeit und Schriftlichkeit“

Table C.1.: Sample duplicates in the Siddata-dataset.

C.3. Results for Classifiers on placetypes

		Alshaikh *et al.* [3] (MDS)					Ager *et al.* [2]					Der. & Sch. [1]		This work		
		Rand.	AHC	Prim.	Sub	Ortho	FTMDS	MDS	FTAWV	AWV	LDA	Best	Best BL	B-all	Avg-B	A-UnW
Four-square	**D1**	0.39	0.36	0.36	0.43	**0.45**	0.41	0.38	0.39	0.32	**0.55**		-	**0.50**	0.45	0.36
	D3	0.50	0.46	0.48	0.54	**0.57**	0.44	0.42	0.42	0.37	0.48		-	**0.58**	0.5	0.54
	DN			-			0.41	0.42	0.41	0.31	0.47	**0.53**	**0.53**	**0.57**	0.55	0.54
	Any			-					-			**0.73**	0.72	-	-	-
Geo-names	**D1**	0.23	0.22	0.24	0.20	0.28	0.32	**0.32**	0.31	0.28	**0.34**		-	**0.51**	0.48	0.49
	D3	0.27	0.29	0.27	0.32	**0.34**	0.31	0.31	0.29	0.28	0.32		-	**0.54**	0.51	0.29
	DN			-			0.24	0.21	0.23	0.22	0.27	**0.37**	0.2	**0.46**	0.44	0.42
	Any			-					-			**0.41**	0.36	-	-	-

Table C.2.: F1-scores of classifiers predicting GeoNames- and Foursquare-labels for three baselines, [1–3] and this work. (long). Described in detail below.

Table C.2 is a longer version of Table 4.1, reporting F1-Scores of classifiers predicting GeoNames- and Foursquare-labels for baselines, [1–3] and this work. **Cls** column encodes the classifier: **D1/3** are DTs of depth 1/3, **DN** an unbounded DT. Condition **Any** refers to the best of all semantic classifiers developed by [1].

First five columns are the exact results reported by [3], next five columns those by [2], afterwards the best config of [1] and the best baseline-condition of [1]. The exact conditions of that work are (left to right, top to bottom): $C4.5_{dir}$, $C4.5_{MDS}$, Col, 1-NN (100D), $C4.5_{dir}$, $C4.5_{MDS}$, $Analog_C$, 1-NN (50D). Explanations of the respective conditions can be found in their work.

All scores are reported with a train-test split of 70% to 30%. Note that the reported results of [2] are unrepresentive when compared to the other datasets: For the placetypes-dataset, **LDA** performs consistently better than their methods, which is not the case for all other datasets used by them. In the case of [3], the results for the placetypes-dataset seem to indicate that the **Ortho**-condition performs consistently better than the **Sub**-condition, which was however not the case for any of the other datasets the authors considered.

Final columns are the results of this work. Column **Best-all** encodes a different for each row, specifically the one that yields the best result for the respective classification task, whereas column **Mean-Best** refers to the single configuration that achieved the best results on average. Last column is best when class weighting of per-class-classifiers is forbidden.

C.4. Comparison of different Cluster-Center-Algorithms

Dimension	reclassify	main
isawyoufirst	beach	beach
workspace	office	
nutrition	restaurant	deli
goalie	stadium	footballstadium
pumperbuilding	county	
starwoodhotels	hotelroom	pool
interstate10	highway	mongolianrestaurant
urban	interior	movietheater
tuolumne	creek	nationalforest
cabs	downtown	
investment	school	stockexchange
stripmall	downtown	departmentstore
michiganstateuniversity	school	campus
ews	railroad	train
anchored	boat	pier
a10	airport	
wc2	restaurant	square
airbase	airport	airbase
joshuatreenationalpark	canyon	
clinker	building	

Table C.3.: Highest-ranking descriptions per dimension for the reclassify-algorithm and the main-algorithm.

C.5. F1-scores per faculty

Depth	1	2	3	unbound
Sozialwissenschaften	0.386 ± 0.029	0.414 ± 0.031	0.403 ± 0.024	0.580 ± 0.034
Kultur-/Geowissenschaften	0.454 ± 0.023	0.514 ± 0.034	0.591 ± 0.026	0.685 ± 0.020
Erziehungs-/Kulturwissenschaften	0.600 ± 0.019	0.701 ± 0.015	0.713 ± 0.019	0.765 ± 0.015
Physik	0.096 ± 0.019	0.117 ± 0.012	0.145 ± 0.023	0.552 ± 0.072
Biologie/Chemie	0.133 ± 0.019	0.178 ± 0.043	0.247 ± 0.063	0.611 ± 0.084
Mathematik/Informatik	0.215 ± 0.032	0.211 ± 0.046	0.260 ± 0.057	0.542 ± 0.070
Sprach-/Literaturwissenschaften	0.652 ± 0.020	0.685 ± 0.014	0.733 ± 0.014	0.790 ± 0.017
Humanwissenschaften	0.177 ± 0.028	0.245 ± 0.061	0.276 ± 0.078	0.484 ± 0.043
Wirtschaftswissenschaften	0.198 ± 0.022	0.219 ± 0.059	0.236 ± 0.040	0.596 ± 0.084
Rechtswissenschaften	0.617 ± 0.108	0.456 ± 0.037	0.641 ± 0.050	0.841 ± 0.032
Mean (weighted)	0.505 ± 0.026	0.553 ± 0.026	0.597 ± 0.026	0.710 ± 0.026
Mean (unweighted)	0.353 ± 0.032	0.374 ± 0.035	0.424 ± 0.039	0.645 ± 0.047

Table C.4.: Robust F1-scores per faculty of a well-performing configuration. The reported results are mean and standard deviation from the result of ten runs with 5-fold crossvalidation each.

C.6. Sample Classification

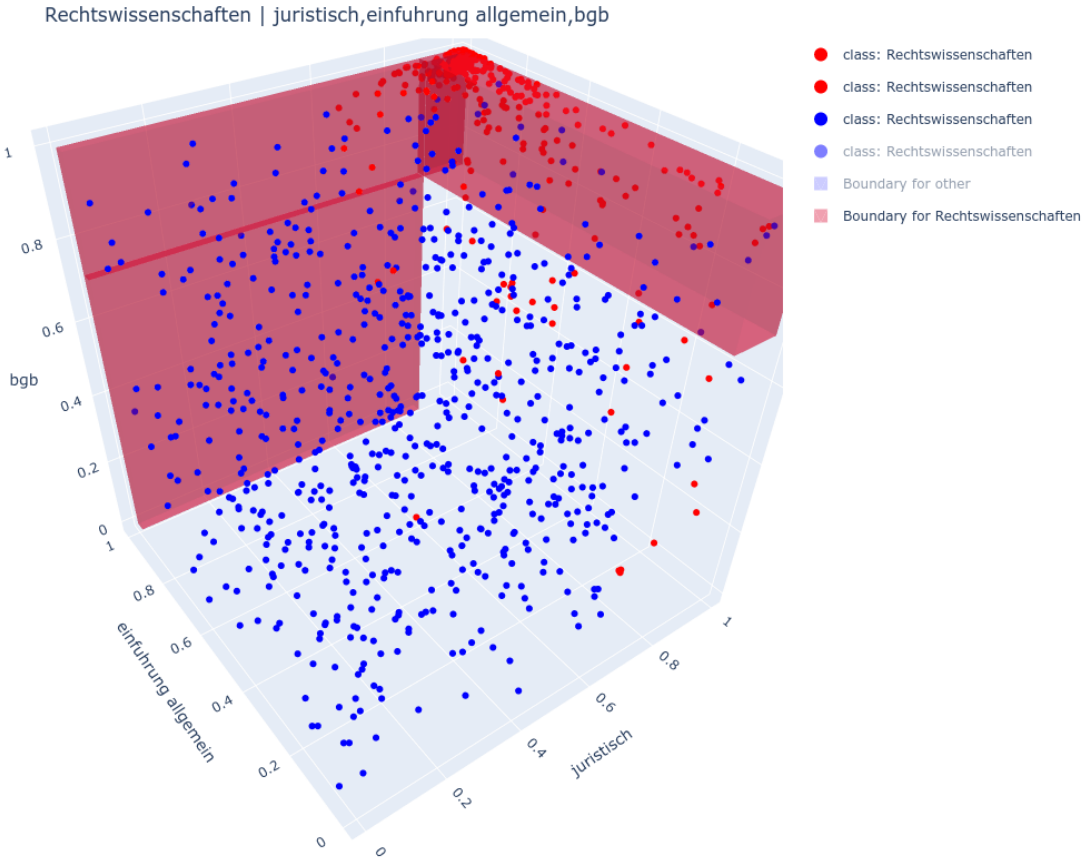

Figure C.3.: Sample classification of a level-2 decision tree. Visualise in 3D: `https://github.com/cstenkamp/derive_conceptualspaces/blob/main/notebooks/text_referenced_plots/display_top3_SIDDATA.ipynb`

C.7. Hyperparameter search

Preprocessing	Quantification	#Dims	Doc-Term-Matrix Quantification	r2r-d	r2r-min	dig	r2r+d	r2r+min	r2r+max	dig+2	c2r+	mean
• Sentence-wise merge • add titles • add subtitles • rm HTML-tags • lower-case • rm stopwords • rm diacritics • use SK-Learn	count	3	ppmi	0	1	0	145	370	510	191	-	174
			tfidf	0	1	0	110	237	278	83	-	101
		100	count	0	5	0	0	114	52	290	0	58
			ppmi	0	6	27	139	224	247	120	-	109
			tfidf	0	6	5	246	270	281	201	-	144
		200	count	0	5	1	0	133	52	509	0	88
			ppmi	0	6	57	196	315	344	90	-	144
			tfidf	0	6	17	357	370	372	433	-	222
	ppmi	3	ppmi	0	0	0	192	247	363	136	-	134
			tfidf	0	0	0	169	206	217	59	-	93
		100	count	0	0	0	0	38	25	242	0	38
			ppmi	0	0	0	80	112	101	22	-	45
			tfidf	0	0	0	89	90	96	85	-	51
		200	count	0	0	0	0	34	21	293	0	44
			ppmi	0	1	112	100	163	163	37	-	82
			tfidf	0	1	127	99	107	106	131	-	82
	tfidf	3	ppmi	0	0	0	229	357	423	84	-	156
			tfidf	0	0	0	169	255	258	24	-	101
		100	count	0	1	0	0	162	64	450	0	85
			ppmi	0	1	3	324	404	423	151	-	187
			tfidf	0	1	0	390	422	437	425	-	239
		200	count	0	2	0	0	211	83	869	1	146
			ppmi	0	2	13	395	559	577	153	-	243
			tfidf	0	2	0	531	554	572	794	-	350
• Sentence-wise merge • add titles • add subtitles • rm HTML-tags • Sentence-tokenisation • lower-case • rm stopwords • Lemmatize • rm diacritics • rm punctuation	count	3	ppmi	0	1	0	226	319	317	208	-	153
			tfidf	0	1	0	210	214	215	82	-	103
		100	count	0	7	0	0	118	61	230	0	52
			ppmi	0	8	27	184	256	262	125	-	123
			tfidf	0	8	5	253	255	255	168	-	135
		200	count	0	8	0	0	117	64	290	0	60
			ppmi	0	11	41	200	319	325	88	-	141
			tfidf	0	11	8	331	333	333	302	-	188
	ppmi	3	ppmi	0	0	0	138	310	321	254	-	146
			tfidf	0	0	0	143	148	150	187	-	90
		100	count	0	0	0	0	29	11	186	0	28
			ppmi	0	1	0	117	142	142	20	-	60
			tfidf	0	1	0	122	124	124	103	-	68
		200	count	0	1	0	0	25	10	272	0	38
			ppmi	0	1	48	126	161	165	28	-	76
			tfidf	0	1	17	143	144	148	133	-	84
	tfidf	3	ppmi	0	0	0	146	219	223	133	-	103
			tfidf	0	0	0	108	111	109	38	-	52
		100	count	0	1	0	0	160	54	389	0	76
			ppmi	0	2	9	281	375	380	205	-	179
			tfidf	0	2	0	373	377	392	339	-	212
		200	count	0	3	0	0	199	64	661	0	116
			ppmi	0	3	21	362	456	472	164	-	211
			tfidf	0	3	1	499	498	501	645	-	307

Table C.5.: Amount of candidates extracted for different parameter-combinations and kappa-scoring-methods. The meaning of the columns is listed in Table B.1. Cells encode the number of terms that exceeded a threshold of 0.5.

D. Algorithm as Pseudo-Code

```
def create_conceptual_space(entities):
    entities = preprocess(entities)
    dtm = create_dtm(entities)
    candidates = dtm.get_words(min_df=25)
    embeddings = mds(dtm)
    good_phrases = []
    for phrase in candidates:
        target = [doc.count(phrase) > 0 for doc in dtm]
        svm.fit(embeddings, target)
        if compare(svm, dtm) > threshold:
            good_phrases.add(svm)
    clusters = cluster_phrases(svm)
    clusters = postprocess(clusters)
    return cs_embedding(entities, clusters)


def preprocess(entities, options):
    for entity in entities:
        # depending on options:
        entity.prepend_title()
        entity.lemmatize()
        # ...
        entity.create_bow()
    return entities

def compare(svm, dtm):
    distances = [svm.distance_to_hyperplane(doc) for doc in dtm]
    counts = [doc.quantification(phrase) 0 for doc in dtm]
    return kappa_score(rank(distances), rank(counts))

def cluster_phrases(svm):
    ...

def cs_embedding(entities, clusters):
    new_embed = [cluster.distance(entity)
                    for cluster in clusters
                        for entity in entities]
    new_embed = rank(new_embed)
    return new_embed
```

Bibliography

1. Derrac, J. & Schockaert, S. *Inducing semantic relations from conceptual spaces: a data-driven approach to plausible reasoning* tech. rep. (2015).
2. Ager, T., Kuželka, O. & Schockaert, S. *Modelling salient features as directions in fine-tuned semantic spaces* in *CoNLL 2018 - 22nd Conference on Computational Natural Language Learning, Proceedings* (Association for Computational Linguistics, Stroudsburg, PA, USA, 2018), 530–540. doi:10.18653/v1/k18-1051.
3. Alshaikh, R., Bouraoui, Z. & Schockaert, S. *Hierarchical linear disentanglement of data-driven conceptual spaces* in *IJCAI International Joint Conference on Artificial Intelligence* (2020), 3573–3579. doi:10.24963/ijcai.2020/494.
4. Schröder, M. *Studienwahl unter den Folgen einer radikalen Differenzierung* 224 (Klinkhardt, Bad Heilbrunn, 2015).
5. Atenas, J., Havemann, L. & Priego, E. Opening teaching landscapes: The importance of quality assurance in the delivery of open educational resources. *Open Praxis* **6.** doi:10.5944/OPENPRAXIS.6.1.81 (2014).
6. Olcott, D. OER perspectives: Emerging issues for universities. *Distance Education* **33,** 283–290. doi:10.1080/01587919.2012.700561 (2012).
7. Ehlers, U.-D. & Kellermann, S. A. *Future Skills - The Future of Learning and Higher education. Results of the International Future Skills Delphi Survey* https://nextskills.org/wp-content/uploads/2020/04/2019-02-23-key-findings-future-skills-report1.pdf. Karlsruhe, 2019.
8. Schurz, K. *et al. TOWARDS A USER FOCUSED DEVELOPMENT OF A DIGITAL STUDY ASSISTANT THROUGH A MIXED METHODS DESIGN* in *18th International Conference on Cognition and Exploratory Learning in Digital Age, CELDA 2021* (2021), 45–52. doi:10.33965/celda2021_202108l006.
9. Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. Analysis of Recommendation Algorithms for E-Commerce (2000).
10. Linden, G., Smith, B. & York, J. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* **7,** 76–80. doi:10.1109/MIC.2003.1167344 (2003).
11. Smith, B. & Linden, G. Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing* **21,** 12–18. doi:10.1109/MIC.2017.72 (2017).
12. Mikolov, T., Yih, W. T. & Zweig, G. Linguistic Regularities in Continuous Space Word Representations. *Proceedings of the 2nd Workshop on Computational Linguistics for Literature, CLfL 2013 at the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2013,* 746–751 (2013).
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. *Distributed representations ofwords and phrases and their compositionality* in *Advances in Neural Information Processing Systems* (Neural information processing systems foundation, 2013). doi:10.48550/arxiv.1310.4546.
14. Le, Q. & Mikolov, T. *Distributed representations of sentences and documents* in *31st International Conference on Machine Learning, ICML 2014* **4** (2014), 2931–2939.

15. Devlin, J., Chang, M. W., Lee, K. & Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* **1,** 4171–4186 (2019).

16. Vig, J., Sen, S. & Riedl, J. The tag genome: Encoding community knowledge to support novel interaction. *ACM Transactions on Interactive Intelligent Systems* **2.** doi:10.1145/2362394.2362395 (2012).

17. Vig, J. *Intelligent tagging interfaces: Beyond folksonomy* in *UIST 2010 - 23rd ACM Symposium on User Interface Software and Technology, Adjunct Proceedings* (2010), 371–374. doi:10.1145/1866218.1866226.

18. Gärdenfors, P. *Conceptual Spaces: The Geometry of Thought* 318. doi:10.7551/mitpress/2076.001.0001 (Bradford Books, 2000).

19. Ai, Q., Azizi, V., Chen, X. & Zhang, Y. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* **11,** 137. doi:10.3390/a11090137 (2018).

20. Mölder, F. *et al.* Sustainable data analysis with Snakemake. *F1000Research* **10,** 33. doi:10.12688/F1000RESEARCH.29032.1 (2021).

21. *ISO / IEC 25010 : 2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models* in (2013).

22. Stockmann, R. & Berg, A. Stud.IP. *eleed* **1** (2005).

23. Gärdenfors, P. How to make the semantic web more semantic. *Formal Ontology in Information Systems. IOS Press,* 17–36 (2004).

24. Fiorini, S., Gärdenfors, P. & Abel, M. Representing part-whole relations in conceptual spaces. *Cognitive processing* **15.** doi:10.1007/s10339-013-0585-x (2013).

25. Schockaert, S. & Prade, H. *Interpolation and Extrapolation in Conceptual Spaces: A Case Study in the Music Domain* in (2011), 217–231. doi:10.1007/978-3-642-23580-1_16.

26. Cohn, A. G., Bennett, B., Gooday, J. & Gotts, N. M. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *GeoInformatica* **1,** 275–316. doi:10.1023/A:1009712514511 (1997).

27. Gärdenfors, P. & Williams, M. A. Reasoning about categories in conceptual spaces. *IJCAI International Joint Conference on Artificial Intelligence,* 385–392 (2001).

28. Dayan, P., Hinton, G. E., Neal, R. M. & Zemel, R. S. The Helmholtz Machine. *Neural Computation* **7,** 889–904. doi:10.1162/neco.1995.7.5.889 (1995).

29. Goodfellow, I. *et al. Generative Adversarial Nets* in *Advances in Neural Information Processing Systems* (eds Ghahramani, Z, Welling, M, Cortes, C, Lawrence, N & Weinberger, K. Q.) **27** (Curran Associates, Inc., 2014).

30. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings.* doi:10.48550/arxiv.1312.6114 (2013).

31. Chen, X. *et al.* InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *Advances in Neural Information Processing Systems,* 2180–2188. doi:10.48550/arxiv.1606.03657 (2016).

32. Blei, D. M., Ng, A. Y. & Jordan, M. I. Latent Dirichlet allocation. *Journal of Machine Learning Research* **3,** 993–1022. doi:10.1016/b978-0-12-411519-4.00006-9 (2003).

33. Schrumpf, J., Weber, F. & Thelen, T. *A Neural Natural Language Processing System for Educational Resource Knowledge Domain Classification* in *DELFI 2021* (eds Kienle, A., Harrer, A., Haake, J. M. & Lingnau, A.) (Gesellschaft für Informatik e.V., Bonn, 2021), 283–288.

34. Dewey, M. A Classification And Subject Index for Cataloguing And Arranging the Books And Pamphlets of a Library. *Search,* 44 (1876).

35. Alshaikh, R., Bouraoui, Z. & Schockaert, S. Learning conceptual spaces with disentangled facets. *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL),* 131–139. doi:10.18653/v1/k19-1013 (2019).

36. Alshaikh, R., Bouraoui, Z., Jeawak, S. & Schockaert, S. *A Mixture-of-Experts Model for Learning Multi-Facet Entity Embeddings* in (Online, 2021), 5124–5135. doi:10.18653/v1/2020.coling-main.449.

37. Kohonen, T. *Exploration of very large databases by self-organizing maps* in *Proceedings of International Conference on Neural Networks (ICNN'97)* **1** (1997), PL1–PL6 vol.1. doi:10.1109/ICNN.1997.611622.

38. Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

39. Lowe, W. Towards a Theory of Semantic Space. *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society,* 576–581 (2001).

40. Firth, J. R. A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis (special volume of the Philological Society)* **1952-59,** 1–32 (1957).

41. Turney, P. D. & Pantel, P. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research* **37,** 141–188 (2010).

42. Bird, S., Klein, E. & Loper, E. *Natural language processing with Python: analyzing text with the natural language toolkit* (" O'Reilly Media, Inc.", 2009).

43. Guo, M. *et al. Effective Parallel Corpus Mining using Bilingual Sentence Embeddings* in *WMT 2018 - 3rd Conference on Machine Translation, Proceedings of the Conference* **1** (2018), 165–176. doi:10.18653/v1/w18-6317.

44. Chen, M. X. *et al. The best of both worlds: Combining recent advances in neural machine translation* in *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)* **1** (2018), 76–86. doi:10.18653/v1/p18-1008.

45. Maas, A. L. *et al. Learning Word Vectors for Sentiment Analysis* in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (Association for Computational Linguistics, Portland, Oregon, USA, 2011), 142–150.

46. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. & Harshman, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* **41,** 391–407. doi:https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9 (1990).

47. Bullinaria, J. A. & Levy, J. P. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods 2007 39:3* **39,** 510–526. doi:10.3758/BF03193020 (2007).

48. Mead, A. Review of the Development of Multidimensional Scaling Methods Author (s): A . Mead Source : Journal of the Royal Statistical Society . Series D (The Statistician) , 1992 , Vol . Published by : Wiley for the Royal Statistical Society Stable URL : https. **41,** 27–39 (1992).

49. Loper, E. & Bird, S. *NLTK : The Natural Language Toolkit* in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics* (Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 2002), 63–70. doi:10.3115/1118108.1118117.

50. Pennington, J., Socher, R. & Manning, C. D. *Glove: Global vectors for word representation* in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), 1532–1543.

51. Nothman, J., Qin, H. & Yurchak, R. *Stop Word Lists in Free Open-source Software Packages* in *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)* (Association for Computational Linguistics, Melbourne, Australia, 2018), 7–12. doi:10.18653/v1/W18-2502.

52. Wartena, C. *A probabilistic morphology model for German lemmatization* in *Proceedings of the 15th Conference on Natural Language Processing, KONVENS 2019* (2020), 40–49.

53. Grootendorst, M. *KeyBERT: Minimal keyword extraction with BERT.* https://doi.org/10.5281/zenodo.4461265. 2020. doi:10.5281/zenodo.4461265.

54. Carmel, D., Roitman, H. & Zwerdling, N. Enhancing cluster labeling using wikipedia. *Proceedings - 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009,* 139–146. doi:10.1145/1571941.1571967 (2009).

55. Breiman, L., Friedman, J., Olshen, R. & Stone, C. *Classification and Regression Trees* 1st Editio, 368. doi:https://doi.org/10.1201/9781315139470 (Routledge, Monterey, California: Wadsworth, 1984).

56. Erk, K. Representing words as regions in vector space. *CoNLL 2009 - Proceedings of the Thirteenth Conference on Computational Natural Language Learning,* 57–65. doi:10.3115/1596374.1596387 (2009).

57. Hernández-Conde, J. A case against convexity in conceptual spaces. *Synthese* **194.** doi:10.1007/s11229-016-1123-z (2017).

58. Shuyo, N. *Language Detection Library for Java* http://code.google.com/p/language-detection/. 2010.

59. Grootendorst, M. *KeyBERT* https://github.com/MaartenGr/KeyBERT. 2021.

60. Miller, G. A. WordNet: A Lexical Database for English. *Communications of the ACM* **38,** 39–41. doi:10.1145/219717.219748 (1995).

61. Hamp, B. & Feldweg, H. *GermaNet - a Lexical-Semantic Net for German* in *Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications* (1997).

62. Henrich, V. & Hinrichs, E. *GernEdiT-The GermaNet Editing Tool* in *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC 2010)* (Valetta, Malta, 2010), 2228–2235.