LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

CENTRUM FÜR INFORMATIONS- UND SPRACHVERARBEITUNG
STUDIENGANG COMPUTERLINGUISTIK

Master's Thesis

in Computational Linguistics

at the Ludwig-Maximilians-Universität München

Faculty of Languages and Literatures

Boosting Performance of a Similarity Detection System using State of the Art Clustering Algorithms

Sabine Ullrich

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

CENTRUM FÜR INFORMATIONS- UND SPRACHVERARBEITUNG
STUDIENGANG COMPUTERLINGUISTIK

Master's Thesis

in Computational Linguistics

at the Ludwig-Maximilians-Universität München

Faculty of Languages and Literatures

Boosting Performance of a Similarity Detection System using State of the Art Clustering Algorithms

submitted by
Sabine Ullrich

Author:	Sabine Ullrich
Supervisor:	Dr Maximilian Hadersbeck
Examiner:	Dr Maximilian Hadersbeck
Work period:	11 March - 12 August 2019

Declaration

I hereby declare that this master's thesis is my own work, I have marked all citations and I have documented all sources and materials used.

Munich, 12 August 2019

. . . .
Sabine Ullrich

Abstract

This thesis strives to boost the performance of the similarity search WiTTSim that finds the most relevant sections in Ludwig Wittgenstein's literary remains to an input query. This is indispensable due to an average query time of 1940.13 seconds, respectively 32 minutes. Therefore, several experiments have been conducted to find the optimal combination of a dimensionality reduction algorithm and a clustering algorithm. Document clustering is deployed as a way of organising the data base beforehand such that the closest, i.e. the most similar, documents can be retrieved in a reasonable amount of time. By selecting the most relevant features by means of Singular Vector Decomposition (SVD) in the pre-processing, the request time can be decreased to 17.19 seconds. By further incorporating K-Means clustering with $k = 150$, the request time can be further reduced to 5.40 seconds. This involves searching the closest cluster centroid for a given query, and subsequently only comparing the datapoints in the most relevant cluster. If the input query is an already known document in the database, a K-Nearest Neighbour search has been implemented to retrieve the most similar documents, yielding high-quality results in only 0.37 to 8.23 seconds.

Diese Arbeit strebt eine Performance-Verbesserung der Ähnlichkeitssuche WiTTSim an, welche die ähnlichsten Abschnitte in Ludwig Wittgensteins Nachlass zu einer bestimmten Suchanfrage ermittelt. Eine Beschleunigung des Prozesses ist zwingend notwendig, da eine Suchanfrage durchschnittlich 1940,13 Sekunden, respektive 32 Minuten, benötigt. Daher wurden mehrere Experimente durchgeführt, um die optimale Kombination zwischen Dimensionsreduktion und Clustering Algorithmus zu finden. Das Clustering der einzelnen Textabschnitte ist dabei eine Vorstrukturierung der Texte, sodass die ähnlichsten Abschnitte in einer angemessenen Zeit ermittelt werden können. Mit Hilfe einer Dimensionsreduktion mittels Singular Vector Decomposition (SVD) kann die Zeit bereits auf 17,19 Sekunden reduziert werden. Mit der Erweiterung eines K-Means Clustering mit $k = 150$, kann die Anfragezeit sogar auf 5,40 Sekunden verringert werden. Dieser Prozess umfasst die Ermittlung des relevantesten Clusters und einem anschließenden Vergleich mit den darinliegenden Datenpunkten. Im Falle einer bereits bekannten Suchanfrage, d.h. wenn das Dokument bereits in der Datenbank vorhanden ist, wurde eine K-Nearest Neighbour Suche implementiert, um die ähnlichsten Textabschnitte zu ermitteln. Dadurch können Ergebnisse von höchster Qualität in nur 0,37 bis 8,23 Sekunden bereitgestellt werden.

Acknowledgements

I would like to offer my special thanks to my supervisor Dr. Maximilian Hadersbeck for his outstanding commitment to encouraging students to get involved in science. The research group around WiTTFind and the Wittgenstein Advanced search tools is truly unique, connecting computational linguistics with the field of philosophy. I highly appreciate to be part of this interesting and interdisciplinary project. Thank you for encouraging me to continue my student research and for the grand opportunity to develop and implement own ideas. Thank you for the constructive feedback and setting me thinking during our regular discussions. I am endlessly grateful to have had such a wonderful supervision and unwavering guidance and support during the entire time.

I am particularly grateful for the assistance by Alois Pichler from the Wittgenstein Archives at the University of Bergen, and for the valuable and expensively hand-labelled evaluation data he provided. His expertise and the constructive feedback have been a great help in developing and evaluating the clustering algorithms.

I would also like to thank everyone at CIS for their kind support and the lively discussions: Dr. Benjamin Roth for his valuable advice concerning useful evaluation criteria, and Daniel Bruder for the discussions and critical questions that were inspiring in all matters. A special thanks goes to Thomas Schäfer for providing disk space and working memory for the calculations.

Above all, I would like to thank my parents with all my heart for their relentless support during my educational career. I am endlessly grateful for all they have done for me, for all the encouragement when I needed it most. I want to thank my sister Monika - her advice and feedback on structuring scientific work have been helpful in all respects. Lastly, I would like to thank my wonderful friends who helped me to escape from my desk to clear my mind from time to time. I coudn't have finished this thesis without knowing all these wonderful people behind me.

Contents

1 Introduction

> *But is a bluish green similar to a yellowish green or not? In certain cases we should say they are similar and in others that they are most dissimilar.*
>
> Ludwig Wittgenstein [100], *Ts-310,87*

Defining similarity is not straightforward and may even be defined differently depending on the given context or situation, as stated by the philosopher Ludwig Wittgenstein in 1934. Analogical to colours, defining the similarity of textual content is equally challenging, and is even more difficult to be specified by machines. However, a vast amout of formal definitions and similarity measures exist that aim to automatically calculate similarities amongst documents [29, 62, 75]. Concerning Ludwig Wittgenstein's Nachlass, searching for similar remarks can be a useful preselection of possible similar remarks, that serves as a basis for new philosphical interpretations.

The introductory chapter will be further subdivided as follows. Section 1.1 will formulate the motivational background of the topic, while Section 1.2 will introduce the scientific aim of this work. The chapter will conclude in Section 1.3 by giving an overview of the entire structure of the thesis.

1.1 Motivation

With the Wittgenstein Nachlass being added to the UNESCO Memory of the World register in October 2017 [117], the electronic accessibility of his literary remains has gained an increased importance for the common documentary heritage. The open source search engine WiTTFind[1] grants electronic access and searchability and further offers several features to allow for a deeper analysis on the philosophical texts by incorporating the Wittgenstein Advanced Search Tools (WAST) [54, 79, 113, 114]. One of the tools that has been developed is the Wittgenstein Similarity search (WiTTSim), a similarity search for retrieving the top k similar remarks to a specified input text [132].

The key challenge of WiTTSim is, however, its brute force approach that simply compares all remarks to each other which leads to a tremendously high complexity and makes the comparison highly inefficient: Retrieving the most similar remarks to a given query q requires approximately 30 minutes which is why WiTTSim does not extend the WAST yet. The long computation time can be explained through the high feature quantity and the large number of approximately 55,000 remarks. Although the Wittgenstein Nachlass is a limited, closed corpus, the incorporated features and the large number of texts to be compared in the sparse vector space make a user-friendly search impossible.

[1]Accessible at `http://wittfind.cis.lmu.de`

To overcome this complexity problem, a sophisticated method for diminishing the search space will be presented. One way of doing so is grouping similar text documents together, also known as document clustering. This method emerged from the field of data mining to extract knowledge from data, but can equally be used to summarise a corpus by grouping together similar topics within a closed corpus [7]. After clustering the documents, only remarks in the most relevant cluster will have to be taken into account which will diminish the search space drastically.

For example: Suppose the query is the text given in Ts-228,136[5][2]:

> Meine Wahl ist frei, heißt nichts anderes als: ich wähle. Und daß ich manchmal wähle, steht doch nicht im Zweifel. Was man "frei" nennt, ist die Wahl. Zu sagen "Wir glauben nur, daß wir wählen" ist Unsinn. Der Vorgang, den wir "wählen" nennen, findet statt, ob man das Resultat der Wahl sich nach Naturgesetzen voraussagen läßt, oder nicht.

The task of the similarity search is to retrieve the top k elements. Instead of comparing 55,000 elements pairwise, the ideal clustering containing 150 clusters is pre-saved and can be used for the comparison. In this example, the cluster closest to Ts-228,136[5] contains 762 remarks. Therefore only 912 similarity pairs will be calculated, 150 comparisons with each cluster centre for retrieving the correct cluster, and 762 points in the retrieved cluster. This reduces the search space by 98.3% and the search time to 5.40 seconds.

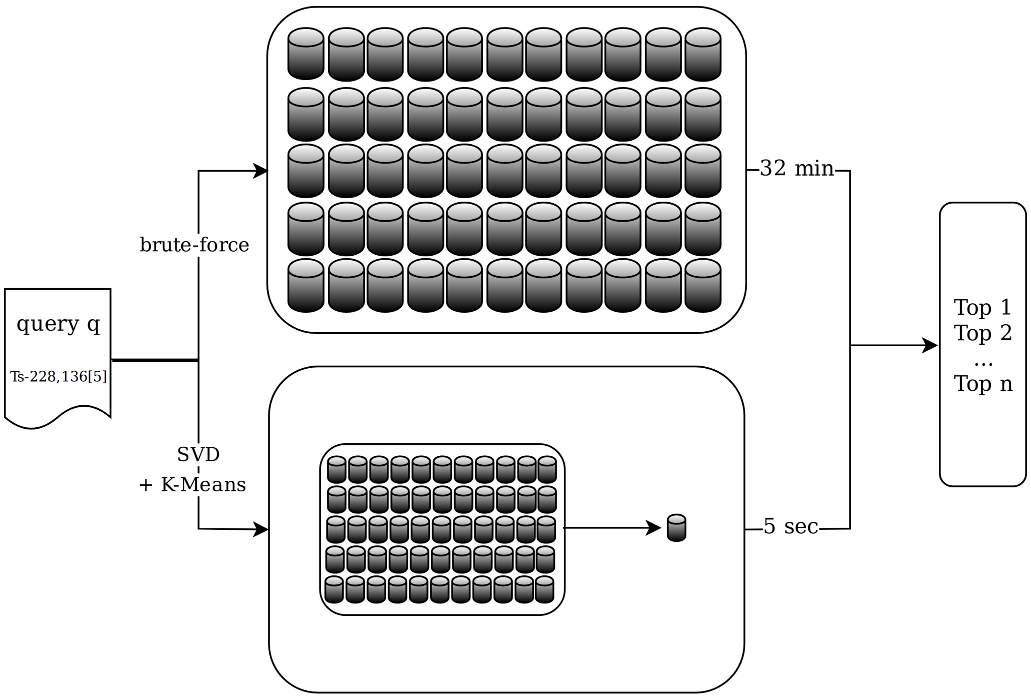


Figure 1.1: Comparison of the brute-force approach and the newly developed methods incorporating dimensionality reduction and document clustering. Each bucket represents 1,000 remarks. By applying SVD and K-Means on the data, the correct cluster is selected and only 912 remarks, i.e. almost one bucket, has to be searched. This reduces the search time from 32 minutes to 5 seconds.

[2]Typescript 228, page 136, remark number 5

1.2 Contribution

A lot of previous work focuses on classifying documents by incorporating a ground truth defined by domain experts. However, the task of clustering is to group similar data points together, which would be redundant if the labels were known beforehand. Therefore, this work only takes into account unsupervised algorithms that can be applied to any realistic setting where no ground truth labels are available.

In the first step, the features were reduced to a reasonable amount. The optimal algorithm for this is found to be Singular Value Decomposition (SVD) which reduces the data in short time and allows for storing the model to be reloaded for the query. In the second step, the reduced data is clustered. Experiments show that ideally this is done by using K-Means with $k = 150$. Being the best fitting combination of reduction and clustering, the methods are then integrated into WiTTSim such that only the relevant clusters will be searched and the search space can be drastically reduced.

Furthermore, two K-Nearest Neighbour (KNN) algorithms were tested. These algorithms map the data systematically to a k-d tree and a balltree respectively, such that a new input query q can be located in the tree and only parts of the tree have to be searched. The performance is compared with the clustering approach and results show that KNN yields excellent results when searching for a document that already exists in the database.

1.3 Outline

The remainder of this thesis is structured as follows. Chapter 2 introduces related work in the field of document clustering. Then, the necessary theoretical background including used terminology and methodology is provided in Chapter 3. Next, Chapter 4 covers the entire implementation of the clustering pipeline, including feature extraction, dimensionaliy reduction, document clustering, integration into WiTTSim, and evaluation techniques used. The experimental results will be presented in Chapter 5, including an evaluation of the dimensionality reduction, the clustering methods and parameters, and the KNN search. The work concludes in Chapter 6 with an overall summary and proposals for future work.

2 Related Work

Over time, an extensive literature has developed on clustering textual documents.[1] A large number of existing studies in the broader literature have developed and tested various clustering algorithms. A closer look to the literature, however, reveals a number of gaps and shortcomings. For instance, most of the clustering algorithms have been applied to artificial textual data where the data points are clearly separated from each other. Also, a vast majority of algorithms have been evaluated in a supervised way. In more natural settings, however, data points are not necessarily separated clearly and in many cases no ground truth labels are available.

In order to get an overview of prior research, this chapter will present existing approaches in the field of document clustering in order to identify useful algorithms that can be applied to our underlying database. Please note that the presentation of papers will be non-exhaustive but will give a good overview of existing research.

The chapter will be further subdivided as follows. Section 2.1 will present partitioning approaches, and Section 2.2 will explain clustering methods using deep learning techniques. Although they could also be classified as deep learning approaches, Kohonen self-organizing maps will be covered separately in Section 2.3. Scientific publications using density-based methods will be introduced in Section 2.4. Other approaches that do not fall in the above listed categories will be outlined in Section 2.5.

2.1 Partitioning Approaches

Paritioning algorithms partition the datasets into groups of similar data points that are closer to each other than to points in other clusters. One of the most widely used algorithms is the K-Means algorithm [64]. K-Means was independently discovered by MacQueen (1967) [83], Ball and Hall (1956) [15], Lloyd (1982) [81], and Steinhaus (1955) [124].

The K-Means algorithm partitions the data into k clusters such that the squared error between the mean of a cluster and the points in the cluster is minimised. Therefore, several parameters have to be specified by the user beforehand. These parameters include the number of clusters k, the cluster initialisations and the distance metric. The cluster initialisations form the initial cluster centroids, i.e. representative points in the cluster centres. Each data point is then assigned to its closest centroid and the centroids are re-estimated. These steps are repeated iteratively until the algorithm converges. A detailed description of the algorithm will be given in Section 3.3.1.

Partitioning approaches are widely used and have been improved over and over. Two common variants of K-Means are ISODATA [15] and FORGY [48]. In fuzzy C-Means [42], the hard clustering from K-Means is replaced by a soft clustering technique where data points can belong to more than one cluster. Fuzzy C-Means has also been further improved

[1]About 411,000 hits in 2009, 1,130,000 hits until 2014, and 2,110,000 hits in 2019 on `https://scholar.google.de/` concerning the clustering of text documents. (Last accessed: 23 July 2019)

over the years [18, 19, 47, 89]. A hierarchical version of K-Means has been presented by Steinbach et al. with their bisecting K-Means algorithm [123]. Pelleg and Moore accelerate K-Means using k-d tree[2] [97] and find a way to automatically determine the parameter k presenting the X-Means algorithm [98]. K-Medoid enables non-numerical data to be clustered [112], while Kernel K-Means [118] allows for clustering nonlinear, arbitrary data.

Practical applications have been presented in various different contexts [7,8,140]. Allahyari et al. [8] present, i.a. the application of partitioning clustering methods in biomedical and health care domains using Latent Semantic Indexing (LSI), Probabilistic Latent Semantic Indexing (PLSA) and topic models. Wei et al. [140] enrich the data beforehand with semantic features to include word sense disambiguation, along with lexical chains to address synonym and polysemy problems. Al-Anazi et al. [7] compare K-Means, K-Means fast, and K-Medoid, along with three different similarity measures namely cosine similarity, Jaccard similarity, and Correlation Coefficient. They find that the best performance is achieved using K-Means and K-Medoids combined with cosine similarity. Further, they state that the resulting clusters improve as the value of k increases. Afonso et al. [3] apply partitioning clustering on graphs representing the Web or social networks, and find that partitioning approaches are particularly time-efficient and yield good results on graph data. For accelerating the clustering progress, Schütze et al. [120] suggest to speed up the distance calculations including LSI and truncation. They find that clustering quality does not suffer while gaining improved performance. However they do not advise to project these efficiency measures onto a similarity search since finding similarities is a lot more fine-grained than classical document clustering.

2.2 Deep Learning Approaches

Xie et al. [145] present Deep Embedded Clustering (DEC), a method that simultaneously learns feature representations in a space Z and a set of k cluster centres using Deep Neural Networks (DNNs). Thereby, a mapping from the data space X to a lower-dimensional feature space Z is required along with a Stochastic Gradiant Descent (SGD) via back-propagation on a clustering objective to learn the mapping. The unsupervised algorithm is designed to be applied to real data, since groundtruth labels for supervision are often not available for hyperparameter cross-validation. DEC runs in two phases: Firstly, the parameters are initialised with a deep autoencoder. Secondly, the clustering parameters are optimised by iterating between computing an auxiliary target distribution and minimising the Kullback-Leibler divergence. This second phase iteratively refines clusters with an auxiliary target distribution derived from the current soft cluster assignment. They compare their own algorithm against K-Means and two spectral-based clustering algorithms (LDMGI and SEC) that use a Laplacian matrix and various transformations to improve clustering performance. DEC shows significant improvements over state-of-the-art clustering methods in terms of both accuracy and running time on two image datasets (MNIST, STL-10) and one textual dataset (REUTERS). It is especially robust with respect to hyperparameter settings which is important when no cross-validation is available in unsupervised tasks.

A convolutional clustering method for unsupervised learning is introduced by Dundar et al. [41] presenting an enhanced version of K-Means clustering. In their work, a deep Convolutional Neural Network (CNN) is trained that combines the strengths of an unsu-

[2]Please refer to Section 3.4.1 for further details regarding k-d tree.

pervised clustering algorithm, K-Means, and CNNs when only few labelled data is available. Thereby, K-Means is modified such that it learns filters that are less redundant at neighbouring locations. Using CNNs along with clustering algorithms has shown immense performance problems after the first layer, also known as the *curse of dimensionality.* The optimised learning algorithm helps to avoid this replication of similar features and extracts only distinctive features from the feature space. Although their method has been tested in the field of image recognition, the convolutional clustering method can also be applied to textual data in a high-dimensional feature space.

Xu et al. [146] tackle the problem of sparsity in clustering short texts. In short texts words often appear only once, hence Term Frequency-Inverse Document Frequency (TF-IDF) will not work. Therefore, they develop a short text clustering system based on deep feature representation, i.e. semantic representations, learned from via Dynamic CNN models. The clue is that this model is an entirely self-taught learning framework which does not rely on any external labels or tags and does not require Natural Language Processing (NLP). For these semantic representations, Xu et al. use the original keyword features and add a locality-preserving constraint. Further, word embeddings, meaning the distributed representation for each word learnt from an external corpus, are taken into account using *word2vec.* They compare the clustering algorithms K-Means, spectral clustering using Laplacian Eigenmaps (LE), and Average Embedding, and find that on search snippets and StackOverflow data, Spectral Clustering and Average Embedding perform significantly better than K-Means.

Another DNN-driven method is presented by Yang et al. [148], introducing an algorithm that combines dimension reduction and K-Means in one step, as can be seen in Algorithm 1. Thereby they map high-dimensional data x_i to its representation h_i in a latent space h where K-Means is suitable for clustering. Subsequently, a loss function measures the reconstruction error.

```
Initialisation{Perform $T$ epochs over the data}
**for** $t = 1 : T$ **do**
    1. Update network parameters;
    2. Update assignment;
    3. Update centroids;
**end**
```

Algorithm 1: The alternating Stochastic Gradient Descent (SGD), presented by [148].

An overview is given in Figure 2.1 where the left side of the “bottleneck” illustrates the forward layers, or encoding, that transform raw data to a low-dimensional space. On the right side, or the decoding, that data is reconstructed from the latent space. The K-Means clustering is then applied to the middle, hence the bottleneck layer in the figure. Both sids are implemented using Rectified Linear Unit activation-based neurons. Yang et al. include an empirically effective initialisation method and an alternating optimisation-based algorithm for handling the cost function in their Deep Clustering Network (DCN). Results show that, compared to many others including basic K-Means, spectral clustering, and DEC, DCN can outperform existing methods in balanced as well as in unbalanced scenarios.

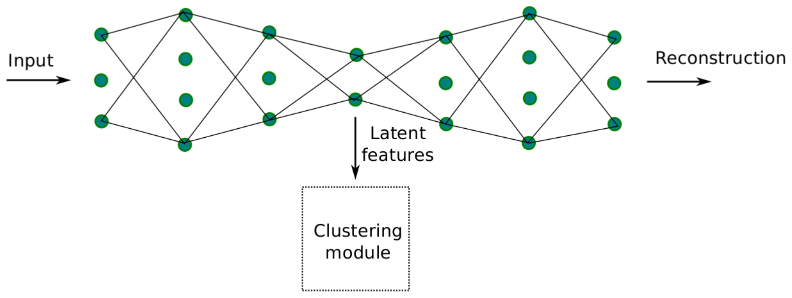


Figure 2.1: Proposed deep clustering network from [148].

Simultaneously learning a DNN along with a clustering algorithm can be improved by learning an embedding space along with subspaces instead of determining the clusters centroid-based [154]. The proposed algorithm can fit every sample into its corresponding subspace and update the subspaces accordingly, even from a bad initialisation [154]. In more detail, k-subspace clustering and convolutional auto-encoder are combined in an end-to-end paradigm where no layer-wise pre-training is required. The updating of subclusters is efficient and easy to scale up to large datasets and results show that cluster accuracy can be increased compared to other algorithms, such as K-Means, DEC, or DCN.

2.3 Kohonen Self-Organizing Maps

The Self-Organizing Map (SOM) is a type of unsupervised feedback neural network that enables data clustering, pattern recognition, information visualization and data mining [153]. It maps high-dimensional input data onto a two-dimensional output grid while preserving the original topology of the data. Methodological details will be given in Section 3.3.5.

The original idea of self organization has been suggested by Willshaw and von der Malsburg in 1976 [142], as well as by Takeuchi and Amari in 1978 [9]. Several years later, Kohonen simplified and optimised their theories and proposed a more practical and robust SOM in 1982 [66,67,153]. Ever since numerous versions, generalisations, accelerated learning schemes, and applications of the SOM have been developed [68].[3]

For example, the SOM-based algorithm WEBSOM allows for organising large document datasets on the web efficiently [60,61,68]. Its idea is to order and organise automatically arbitrary textual document collections and enable their interactive browsing and exploration [60]. The first step is to encode the texts which is done by histograms retrieved from so-called “self-organizing semantic maps” [108]. They describe the word context in form of vectors and can therefore be compared to word embeddings that enable a consideration of synonymous words. In the next step, results are blurred with a Gaussian convolution kernel to adjust small invariances and the encoding is saved to a hash table for performance reasons. The resulting SOM reflects the relations between i.a. newsgroup articles, and other types of similar articles, but can also applied to larger web pages or email messages. In WEBSOM, categories may overlap, i.e. a web page can address multiple topics.

[3]A comprehensive list of 5,384 scientific papers that include SOMs in their research has been collected by Oja et al. [91] and can be accessed at `http://www.cis.hut.fi/nnrc/refs/`.

Bação et al. [12] find that a SOM is even less prone to local optima and that the search space is better explored than in a classical K-Means clustering. To motivate the choice of SOMs over neural networks, Kangas et al [65] introduce two variants that exceed the results of conventional methods for clustering and classification. Firstly, they present a dynamic weighting algorithm that weighs the input data of each input cell for improving the ordering when the input data is highly unstructured. Secondly, they define the neighbourhoods in the learning algorithm by means of a minimal spanning tree, which assigns arcs between the nodes so that all nodes are connected through single linkages. Moreover, the total sum of the lengths of the arcs is minimised, which provides a better and faster approximation of distinctive structured density functions.

In order to get good insight into the cluster structure, a SOM can be applied as an intermediate step before performing any other clustering method on the data, such as hierarchical agglomerative clustering or partitive clustering. Therefore, the SOM produces the prototype set and the second stage produces the actual clustering. This intermediate step helps to effectively visualise and explore properties of the data [134]. Furthermore, SOMs can have additional applications in other NLP fields, such as the improvement of speech recognition and word sense disambiguation [59].

2.4 Density-based Approaches

Ester et al. [43] present a formal model for density-based clusters, as well as a database-oriented algorithm Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to find clusters that adhere to this model. Therefore, they apply a simple minimum density level estimation based on a threshold for the number of neighbours *minPts* within a specified radius ϵ. Objects with more than the number of *minPts* neighbours within ϵ are considered to be a core point [119]. With this method, DBSCAN finds clusters that satisfy the minimum density, separated by areas of lower density [119]. The methodological details of this appraoch will be further explained in Section 3.3.3.

Their algorithm has been adapted and improved ever since [25, 27, 69]. For instance, the HDBSCAN (Hierarchical DBSCAN) [85] algorithm abandoned the concept of border points, and considers only core points to be part of a cluster at any time. It defines, along with LSDBC (Locally Scaled Density Based Clustering) [20], versions of DBSCAN variants for finding hierarchical clustering results.

Another variant of DBSCAN is OPTICS (Ordering Points To Identify the Clustering Structure), which does not produce a clustering explicitly. Instead, several distance parameters are processed at the same time and the density-based clusters with respect to different densities are constructed simultaneously [10].

Lastly, DenClue (DENsity-based CLUstEring) [58] considers alternative kernels for density estimation. It should be pointed out that for any of the density-based algorithms the choice of a suitable value of ϵ is indispensable, i.e. ϵ should not be extremely high or low, and in general, extreme parameter settings should be avoided. Further, evaluation should be done with respect to the utility of the resulting clusters [51, 119].

2.5 Other Approaches

Other approaches not included above comprise for example hashing-based clustering techniques, which are especially useful for large amounts of data and are widely used for decreasing complexity in the calculation of similar documents [31, 104, 122, 152]. This is mainly done by using Locality-Sensitive Hashing (LSH) which can reduce the complexity of n data points with k features from $O(n^2k)$ to a complexity proportional to $O(nk)$ [104,152]. The idea of LSH has been introduced by Broder [24] and bases on the use of hash functions for fingerprinting of very large strings by Rabin [102].

LSH is a randomised and probabilistic algorithm and works as follows: Represent the n number of data vectors in a d-dimensional space. Then generate a random number k of hyperplanes and assign binary values to the hash value of each data point depending on which side on the hyperplane it is located. In other words, each data point in the vector space is represented by a hash value of length k (see Table 2.1 as hash table for Figure 2.2). Its goal is to have similar hash codes for nearby points, hence points with similar hash values should be close in the vector space. Repeat these steps l times to get l different hash tables. Incorrect hash values, are eliminated by comparing the different hash tables.

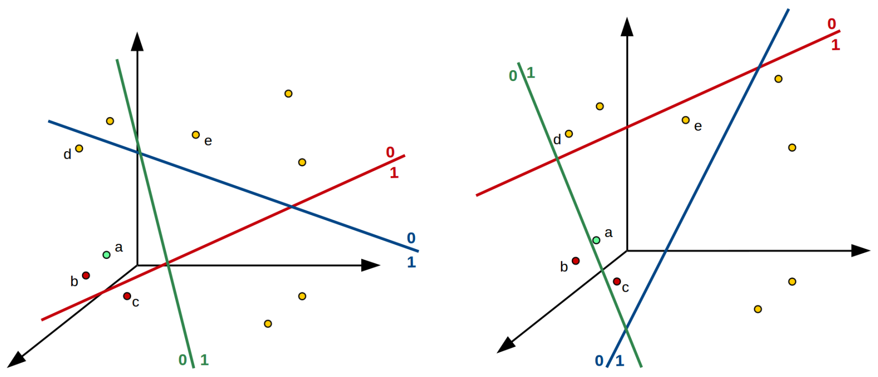


Figure 2.2: Separating the vector space by randomly chosen hyperplanes and defining binary values for the hash value of each data point.

Hash 1	Cluster 1	Hash 2	Cluster 2
100	abd	001	d
001	e	010	b
110	c	001	ace
..		..	

Table 2.1: Calculating the hash values for Figure 2.2. The first value reflects, which data points lie on the side 1 of the blue line, and 0 of the red and green line, yielding the cluster a, b, and d. The remaining values are determined equivalently.

An efficient variant of LSH is SimHash [115] where hyperplanes are not stored but recreated on-the-fly. After the preprocessing, i.e. tokenizing, stopword removal and so forth, a unique b bit binary hash for every word is computed. Then, all zeros are converted to -1 and multiplied by the word weight, which is usually done using TF-IDF weighting. Each of the columns is summed up and set to 1 if the sum is greater than zero, and set to 0 otherwise. This creates the vector V which forms again the basis of the document fingerprint.

Conventional unsupervised clustering methods partition data points in an n-dimensional space without any relationship information given. However in some cases, additional information can be provided, such as cluster labels of some observations, or certain data points may be known to belong to the same cluster. In these cases, semi-supervised clustering methods can be applied to partially labelled data which yields better results than applying supervised data classification on datasets where only a small set of labelled data is provided [13]. For example, labelled instances can be integrated into the classical K-Means algorithm, also known as constrained K-Means clustering [13]. In that case, labelled observations are always assigned to their known cluster even if they are located closer to the mean of another cluster (see Figure 2.3). As a variant, seeded K-Means clustering [16] uses the labelled data only for the initial seeding step. Therefore, the labelled data provides information for the choice of the initial cluster centres, while the following steps do not differ from the classical K-Means algorithm. Other semi-supervised clustering methods comprise i.a. the use of kernels [11, 45], silhouette capturing [155], and direct interaction with domain expert feedback [44, 72].

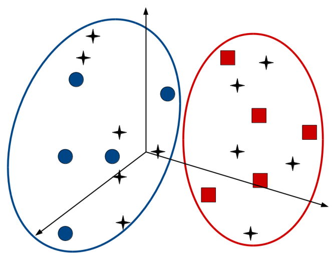


Figure 2.3: Semi-supervised approach to clustering using the constraint K-Means algorithm. The blue circles and red squares represent labelled data. The black stars show unlabelled data. All data points are clustered with respect to the predefined class labels.

A preprocessing step based on word frequency is presented by Patki et al. [94], suggesting a custom feature selection method based on documents where each feature is equivalent to one word. Firstly, unfrequent words are disregarded, i.e. features are only considered when they occur in at least k documents, where k depends on the number of documents to be clustered. Secondly, short words are excluded from the feature vector which means only features whose length $l \geq 3$ are considered.

Especially for high dimensional data spaces it can be useful to examine subspaces of the original dataspace separately. In doing so, relevant features can be unveiled and clusters can be extracted. Therefore, the data has to be separated into subsets and the most relevant dimensions determined [135]. It has to be noted that the retrieved clusters are possibly located in overlapping subspaces [93]. Subspace clustering is divided into top-down search and bottom-up search. The former includes iterative methods where the initial clustering is performed onto the full set of dimensions and then the subspaces of each cluster evaluated. A large number of top-down approaches have been presented, including PROCLUS [4], ORCLUS [5], FINDIT [144], δ-Clusters [149], and COSA (Clustering On Subsets of Attributes) [50]. Bottom-up searches on the other hand are grid-based methods, that means, dense regions in low dimensional space are identified and combined to form clusters. Algorithms that involve on bottom-up subspace clustering are i.a. CLIQUE [6], ENCLUS [32], MAFIA [52], Cell-based Clustering (CBF) [30], CLTree [80], and DOC (Density-based Optimal projective Clustering) [101].

3 Methodological Overview

In this work, several clustering techniques will be presented and evaluated for boosting the performance of WiTTSim. In order to understand the implementation process and the analysis presented later, a short prelude discussing the underlying methodology and utilised technical terminology is necessary. A general overview of a common clustering pipeline is illustrated in Figure 3.1. The pipeline comprises a translation from documents into vectors, a feature selection and selection of clustering algorithm, a validation of clusters and interpretation, and possible re-selection of algorithms. Subsequently, knowledge is gained from the clustering outcome and the search room can be restricted.

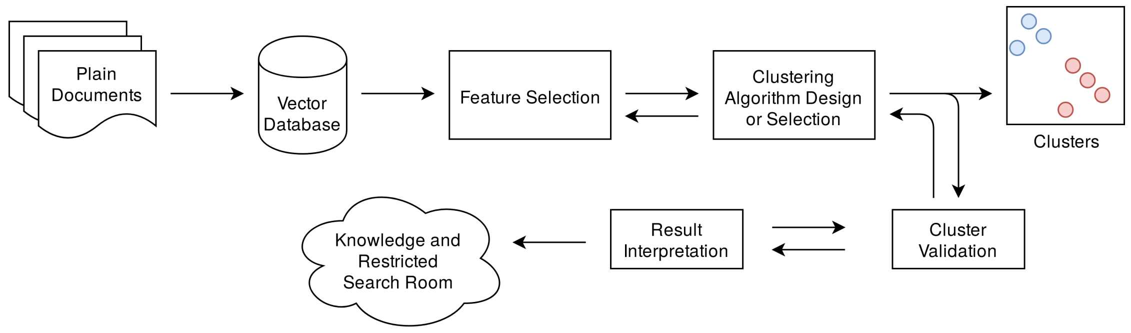


Figure 3.1: Pipeline for Document Clustering adapted from [147].

This chapter is subdivided as follows: Section 3.1 will overview the feature extraction of the underlying database. In Section 3.2, dimensionality reduction methods are presented as a crucial preprocessing step of clustering high-dimensional data accurately. Section 3.3 will explain various types and algorithms in the field of document clustering in more detail. Further, an efficient organisation algorithm based on KNN trees will be presented in Section 3.4, which may also be applicable in boosting performance of WiTTSim. The chapter concludes in Section 3.5 with an overview of common evaluation methods for document clustering.

3.1 Feature Extraction

For the feature extraction, each of the preprocessed documents is translated into binary feature vectors, which has been implemented in [132]. This means that the documents can be represented thereafter in the Vector Space Model (VSM), where each dimension corresponds to one feature of the input vector. The features that have been taken into account are the words themselves, the lemmas, the Part-of-Speech (POS)-Tags, and the respective synonyms. For extracting the corresponding synonyms of the words, GermaNet [55, 57] has been employed in the feature extraction. GermaNet is a lexical-semantic net that groups lexical units such as nouns, verbs, and adjectives with similar semantic concepts into so-called *synsets* by defining their semantic relations. The equivalent synonym extraction for the English language has been implemented using WordNet [46, 88].

Technically the document representation can be defined as follows: Let $\mathcal{D} = d_1, d_2, ..., d_M$ be the collection of M documents where each document d is translated into an n-dimensional vector in the VSM. Each position of the vector represents different characteristics $\mathcal{C} = c_1, c_2, ..., c_n$ of the text. As explained above, these features comprise the words themselves, the lemmas, POS tags, and synonyms. The number of occurence of these features, i.e. their frequency $f_d(c)$, is also denoted in the vector for each document. The symbolical vectors for two of the documents are depicted in Table 3.1.

remark	w_1	w_2	...	w_m	l_1	l_2	...	l_n	p_1	p_2	...	p_o	s_1	s_2	...	s_p
Ms-101,IIr[1]	1	0	0	1	1	1	0	0	0	1	0	0	1	1	0	1
Ms-101,IIr[2]	0	1	0	1	0	0	0	1	0	0	1	1	1	0	0	0
...	0	1	...	1	0	...	0	1	...	0	1	...	1	0	...	...

Table 3.1: Symbolical vector structure for two remarks in Manuscript 101 – Ms-101,IIr[1] and Ms-101,IIr[2] – where each vector stores the feature information regarding the occurrence of a word (w), its lemma (l), POS-tag (p), and respective synonyms (s). (Table adapted from Tan et al. [127])

The most similar documents, i.e. the closest vectors in the n-dimensional vector space, are determined by computing the cosine distance between the vectors. Since the vector dimension n is tremendously high for the underlying database and because of the large number of remarks to be compared, all vectors are computed in advance. This step is only possible, since the search space is closed and no further documents are added dynamically. However, the large number of features leads to severe problems for the clustering process, as high-dimensionality leads to data sparsity where all data points are far from one another. This is why a dimensionality reduction method will be essential, as described in the following section.

3.2 Dimensionality Reduction

An important step in data mining is the task of dimensionality reduction. With an increase of dimensionalities, clustering becomes significantly more complex and more time-consuming compared to clustering data in lower dimensional space. While a larger number of dimensionalities allows to store many input variables, not all intuitions developed in spaces of low dimensionality will generalize to spaces of many dimensions, also known as the *curse of dimensionality* [23]. In sparse high-dimensional spaces, reducing dimensionalities by eliminating redundant or irrelevant features increases the performance of clustering, which can either be achieved in a supervised or unsupervised way. This means that for each document vector with length n, eliminate i unnecessary features in order to find the minimum set of attributes such that the resulting probability distribution of the clusters is as close as possible to the original distribution [56, 78]. The new vector space is then of $k = n - i$ dimensions and reduces the space from $\mathbb{R}^n$ to $\mathbb{R}^k$.

Various approaches exist for mapping the higher dimensional space to a lower dimensional space. Depending on the information base, either supervised or unsupervised approaches can be taken into account. Supervised methods can be applied to data where the classes are known in advance, and the reduction can learn relevant features from the labelled data basis. A well-known algorithm is for instance Linear Discriminant Analysis (LDA). Contrarily, unsupervised reduction methods attempt to reduce the dimensionality by retaining as much information as possible without any ground truth labels provided. Two of most popular unsupervised approaches are SVD and Principal Component Analysis (PCA) [78]. The most significant concepts of dimensionality reduction will be presented in the following sections.

3.2.1 Singular Value Decomposition

SVD is a generalization of the eigenvalue decomposition. It states that any data matrix $M = m \times n$ in $\mathbb{R}^{m \times n}$ with m objects and n dimensions can be decomposed such that $M = U\Sigma V^T$ where U is a $m \times r$ column-orthonormal matrix, Σ is a diagonal $r \times r$ matrix, and V is a $r \times n$ column-orthonormal matrix, see Figure 3.2.

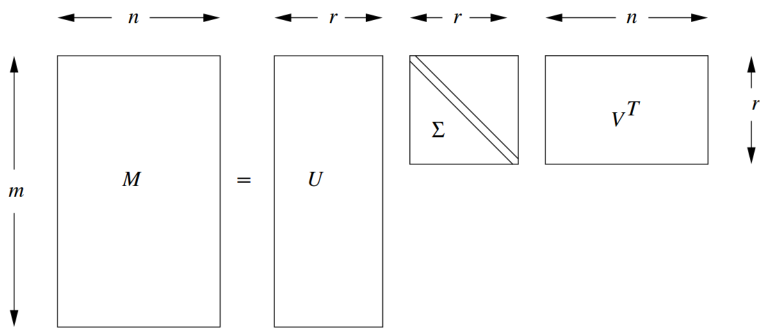


Figure 3.2: The decomposition used in SVD, taken from [131]. The first rectangle represents the original space $M = m \times n$ with m objects and n dimensions. The space can be partitioned into $M = U\Sigma V^T$ where the singular values in Σ are set to U such that the corresponding columns in U and rows in V^T can be eliminated.

In order to reduce the number of dimensionalities, the smallest singular values in Σ are set to U and the corresponding columns in U and rows in V^T are eliminated. The number of dimensions to be eliminated depends on the result accuracy. That is to say, enough singular values should be retained, such that at least 90% of the energy in Σ is preserved. In data spaces with a large number of dimensionalities, SVD can yield more accurate results than other algorithms. This is because of the data sparsity in large dimensions where determining the largest covariance, as required by other methods, can be problematic. [71]

3.2.2 Principal Component Analysis

Based on SVD, PCA is similarly a linear dimensionality reduction method, first presented by Pearson [95]. Although various dimensionality reduction methods exist, PCA is the most popular one [133]. Its goal is to find k dimensions that best fit onto the data and that maintain a reasonably high variance in the data.

PCA projects m data points $x_1, x_2, ..., x_m \in \mathbb{R}^n$ from $\mathbb{R}^n$ to a lower dimensional space $\mathbb{R}^k$ by computing the following steps: First, the data is normalized to have a mean of 0 and a standard deviation of 1. Then, the covariance matrix is computed by Equation (3.1) [33, 76]:

$$\Sigma = \frac{1}{m} \sum_{i=1}^{n} (x_i)(x_i)^T \tag{3.1}$$

where T is the transposed vector (that is $n \times 1 \rightarrow 1 \times n$).

After obtaining the covariance matrix, compute the k orthogonal principal eigenvectors of Σ $u_1, ..., u_k \in \mathbb{R}^n$, which are the orthogonal eigenvectors of the k largest eigenvalues. Lastly, the eigenvectors are sorted as per increasing eigenvalue to maximize the variance among all k-dimensional spaces. An illustration for finding the maximum variance is given in Figure 3.3.

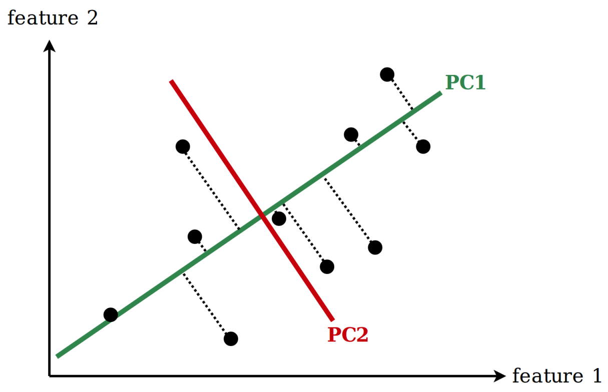


Figure 3.3: Determining the maximum variance among the data points in PCA. The first principal component (PC1) represents the axis with the highest variance. The second principal component (PC2) is orthogonal to PC1.

An extension of PCA is the Independent Component Analysis [34] which imposes independence up to the second order and utilises Mutual Information (MI) as a function of cumulants of increasing orders. Since data points in a feature space are not necessarily linear, the methods above can quickly become problematic in many cases.

Therefore, Kernel PCA is introduced, which is another modification of PCA that can be applied to non-linear data [71, 133]. Its idea is to implicitly create a linear space using the so-called *Kernel trick*: thereby a kernel function

$$K(x, y) = \phi(x)^T \phi(y) \tag{3.2}$$

which is the scalar product of the two matrices, is calculated pairwise. The function ϕ is mutable and stands e.g. for the degree d polynomials:

$$K(x, y) = (x^T y + c)^d \tag{3.3}$$

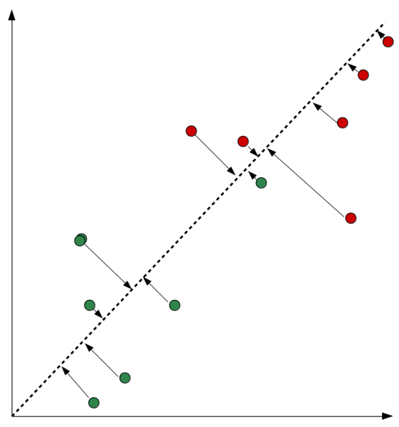
3.2.3 Linear Discriminant Analysis

Instead of maximising the variance of the data points, LDA maximises the separability among known categories [14, 130, 150]. This is done by inserting a new axis and projecting the data onto it.

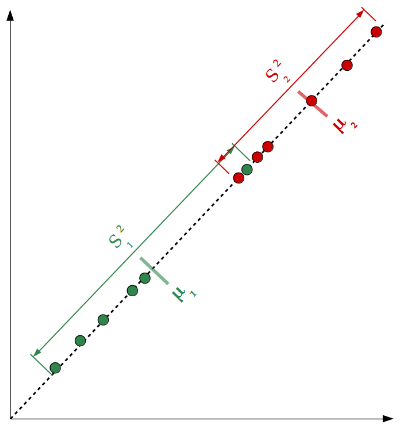
In order to find the ideal new axis, two criteria have to be fulfilled. Firstly, the distance between the means μ_i of the classes has to be maximized. Secondly, the variation within each class s^2, also known as *scatter*, has to be minimized within each category. Equation (3.4) shows how to combine these two criteria for two classes. [14]

$$\frac{(\mu_1 - \mu_2)^2}{s_1^2 + s_2^2} \tag{3.4}$$

An illustration of the algorithm is given in Figure 3.4, depicting the projection of a two dimensional data space onto a new axis. The two mean values along with the scatter for each class are also marked.



(a) Finding the best fitting axis and mapping the data points onto it.



(b) Projecting the data points onto the new axis and maximising the distance of the means μ_1 and μ_2, while minimising the scatters s_1^2 and s_2^2.

Figure 3.4: Linear Discriminant Analysis for dimensionality reduction from $\mathbb{R}^2$ to $\mathbb{R}$.

3.2.4 Sparse Random Projection

In Random Projection (RP) [1,2,77], the original high-dimensional data is projected onto a random lower-dimensional subspace using a random matrix whose columns have unit lengths. The original set of N d-dimensional data points is projected onto a k dimensional subspace with $k < d$ using a random $k \times d$ matrix R, as shown in Equation (3.5). [22]

$$X^{RP}_{x \times N} = R_{k \times d} X_{d \times N} \tag{3.5}$$

RPs are based on the Johnson-Lindenstrauss lemma [36], that proves that points in high dimensional Euclidean space can be mapped into a lower dimensional Euclidean space such that the distance between the points can be nearly preserved. Using random projetions is computationally significantly less expensive than other linear reduction techniques that have to expensively compute the orthogonal axes. However, in high-dimensional spaces, a much larger number of *almost* orthogonal directions than orthogonal directions exists. As a specialised version of RP for sparse data regions, Sparse Random Projection (SRP) is not only computationally efficient, but also yields sufficiently accurate results for reducing high-dimensional datasets. [22]

3.2.5 Uniform Manifold Approximation and Projection

As opposed to the linear techniques presented above, Uniform Manifold Approximation and Projection (UMAP) [86] is a manifold learning technique [38] for dimension reduction that incorporates ideas from topological data analysis. It works similarly to other manifold techniques such as t-distributed Stochastic Neighbour Embedding (t-SNE) [82], however, it performs significantly better at maintaining not only the local but also the global structure of the dataset.

The reduction is computed in two steps. Firstly, the nearest neighbours are extracted efficiently using RP trees [35] and nearest neigbour descent [40]. After constructing the weighted k-neighbour graph, a low dimensional layout of this graph is computed by optimising the layout subquadratically with SGD [87] and negative sampling [128]. Further, UMAP offers supervised dimension reduction, computations among differently combined metrics, and a combination of different datasets.

3.3 Document Clustering Methods

Once the documents are reduced to a reasonable amount of features, the clustering algorithm can be applied to the data. Prior to introducing the clustering techniques, the terminology is defined in the following, according to [78]:

Definition 3.3.1 A *cluster* is a group of objects that are located at the same region.

Definition 3.3.2 The *clustering* process means grouping a set of data objects into clusters, i.e. collections of data objects, without demanding predifined class labels.

According to the definition, similar objects are in similar clusters, while dissimilar objects do not share the same cluster. As opposed to classification, clustering is an unsupervised task, since there are no ground truth labels available. Therefore, the instances are not divided into predicted classes, but are rather divided into natural groups that bear a stronger resemblance to each other than to objects in other groups [143]. This makes the evaluation less trivial than evaluating any classification method. Furthermore, the number of clusters can highly vary depending on the data, while this number is most likely not known in advance [121]. Moreover, groups can either be exclusive, i.e. each element belongs to exaclty one cluster, overlapping, i.e. each element can belong to more than one cluster, or hierarchical, i.e. the top level gives a rough division while each group is refined further [143]. Since the goal of this thesis comprises the calculation of similarity among datapoints, a definition for similarity-based clustering is provided in the following as defined by [73].

Definition 3.3.3 Given is a data set $\mathbf{D} \in \mathcal{L}_e$, a similarity function $sim : D \times D \to \mathrm{R}$ and a quality criterion q where $\mathcal{L}_e$ is the language of the examples e. The task of the *similarity-based clustering* is to find a set of clusters $\mathbb{K} \subseteq \mathrm{P}(\mathbf{D})$ that maximizes the given quality criterion q or minimizes the given error criterion er. For the criterion q, measures comprise i.a. the average intra-cluster similarity that should be maximised, and the average inter-cluster distance that should be minimised.

A classification of clustering types and approaches is provided in Figure 3.5. Each of the methods therein will be further explained in the following subsections.

3.3.1 Partitioning Methods

One of the most widely used partitioning algorithms, due to its simplicity and effectiveness, is the K-Means algorithm, first published by Lloyd in 1982 [81]. K-Means divides data points iteratively into k clusters using centroids as representative points. The centroid is the mean value of the data points within one cluster. Its goal is to minimise the total squared distance from all points to their cluster centres [143]. Thereby the number of clusters k has to be determined in advance and should be chosen thoroughly, since it has a high impact on the overall performance. A poorly chosen value can lead to unsatisfying

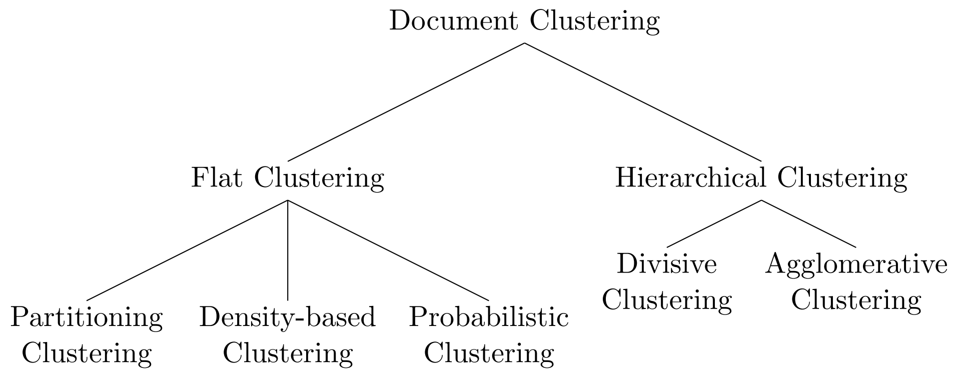


Figure 3.5: Classifying document clustering approaches into flat clustering and hierarchical clustering methods.

results, and the risk of the algorithm converging to a local maximum only, is high [3]. The pseudo code for K-Means is presented in Algorithm 2.

```
input  : $k$; $\mathbf{D}$
output: $\mathbb{K}$
arbitrarily choose $k$ examples from $\mathbf{D}$ as the initial representative examples
 (called *seeds*)
repeat
    1. (re)assign each object to the cluster to which the object is the most
       similar, based on the mean value of the objects in the cluster;
    2. update the cluster means, i.e. calculate the mean value of the objects for
       each cluster;
until *no change*;
```

Algorithm 2: Algorithm for classical K-Means clustering, adapted from [156].

An illustration of the algorithm is provided in Figure 3.6: Firstly, k data points are arbitrarily selected as cluster centroids. Then, the closest points to these centroids are assigned to be in the same cluster. The centroids are recalculated based on the newly assigned points such that they present the new centre of the cluster. These two steps are repeated iteratively until convergence is reached [7, 8].

A major drawback of K-Means is that the final clusters are immensely sensitive to the initial centroid choice. The algorithm can be drastically improved by selecting the initial seeds carefully, as done in the K-Means++ algorithm. There, the first seed is chosen randomly with a uniform probability distribution. Then the second seed is chosen with a probability that is proportional to the square of the distance to its predecessor. At each stage the next seed with a probability proportional to the square of the distance from the closest seed that has already been chosen is defined. This improves both speed and accuracy compared to the arbitrarily selected seeds in the basic K-Means algorithm. [143]

Another drawback is that semantic data is often not numerical, and similarity and dissimilarity should be employed rather than distance. However, many algorithms are designed to cluster interval-based, i.e. numerical, data where a centroid represents a cluster.

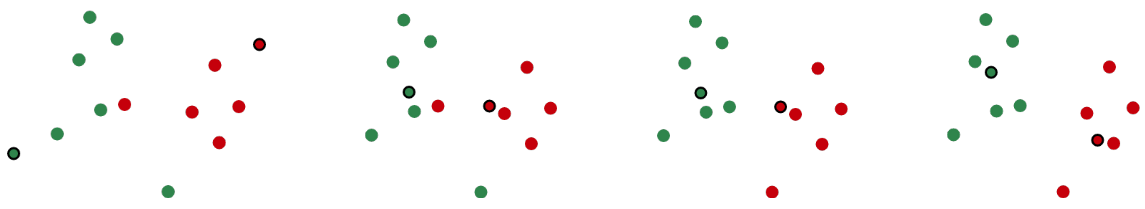

(a) Start (b) Update (c) Reassign (d) Update

Figure 3.6: K-Means overview: Initially, k random seeds are chosen in the dataspace. Each of the data points is assigned to its closest seed. Then, the seeds are updated such that they form the cluster centres of their respective clusters. Next, the data points are reassigned to the new centres. These steps are repeated iteratively until the algorithm converges.

For categorical data, a Partitioning Around Medoids (PAM) algorithm, also referred to as K-Medoid, can be applied instead [73, 121]. Therefore, actual data points represent clusters directly, instead of incorporating centroids as representatives. This makes the algorithm less sensitive to outliers than the classical K-Means algorithm. Algorithm 3 describes the implementation for K-Medoid. Equally, the input k is the number of clusters, $\mathbf{D}$ is the dataset of n examples and output $\mathbb{K}$ a set of k clusters.

```
input  : $k$; $\mathbf{D}$
output: $\mathbb{K}$
arbitrarily choose $k$ examples from $\mathbf{D}$ as the initial representative examples
  (called seeds)
repeat
    1. assign each remaining example $e$ to the cluster $K_i$ that contains the
      most similar representative example $medoid_i$;
    2. randomly select a nonrepresentative example $e_j$;
    3. compute the total cost $er$ of swapping representative example
    $medoid_i$ with $e_j$;
    if $e > 0$ then
        swap $medoid_i$ with $e_j$ to produce a new set of $k$ representative objects;
    end
until no change;
```
Algorithm 3: Algorithm for PAM, presented by [73].

Since K-Means only can be used to cluster numerical data, K-Median and K-Mode can be equivalently applied to ordinal and categorical data. A short comparison of all three algorithms and their characteristics is listed in Table 3.2. The respective algorithm is chosen depending on the type of the underlying dataset.

Since choosing the number of clusters is crucial for the presented approaches, a couple of methods for determining the ideal cluster size will be given. The first possibility to determine the ideal number of clusters is *trial and error*. That is, start from a given minimum, e.g. $k = 1$ and work to a small fixed maximum. Evidently, the best results are achieved in the trivial scenario where each document constitutes of its own cluster, i.e.

	K-Means	**K-Median**	**K-Mode**	**K-Medoid**
data	numerical (mean)	ordinal	categorical	metric
efficiency	high $\mathcal{O}(tkn)$			low $\mathcal{O}(tk(n-k)^2)$
sensitivity to outliers	high	low		

Table 3.2: Comparison of the partitioning methods, presented by [121].

when k equals the number of data points. To tackle this issue, solutions with too many clusters, meaning clusters that are overfitting to the given data, should be penalised [143]. A commonly applied method is determining the Minimum Description Length (MDL), which was presented in 1978 by Rissanen [106, 107]:

$$L(T) + L(E|T) \tag{3.6}$$

where L is the length, T the chosen theory, E the collection of class labels in the training set, $L(T)$ the number of bits for encoding, and $L(E|T)$ the number of bits given a certain theory. MDL proposes to select the T that minimises the sum in Equation (3.6). For clustering, MDL can be especially helpful for the evaluation of clustering and to prevent a system from preferring overfitted clustering models. This helps in the decision whether the result of learning proves useful in the application context or not.

The evaluation can be performed as follows. Divide the training set E into k clusters, measure the averages and distances and encode the cluster centres, i.e. the average value of each attribute over all instances in the cluster. The best clustering will support the most efficient encoding E. In other words, the description length will decrease for a strong clustering result compared to a bad clustering result. [143]

A last possibility is to find clusters e.g. $k = 2$ and determine whether it is worth splitting them or not. For example, create a new seed one standard deviation away from the cluster's centre in the direction of its greatest variation and create a second seed the same distance in the opposite direction. Decide with one of the methods from Table 3.3 on page 26 if the split should be retained, if so, proceed iteratively. [143]

3.3.2 Probabilistic Methods

This section will describe and explain statistical approaches to document clustering. Similar to partitioning-based methods, probabilistic methods involve a recalculation and reassigning of objects to improve parameters iteratively. The result is a soft clustering outcome where each data point is assigned to all clusters with a certain probability. It can be extended and artificially adapted by choosing the cluster that yields the highest probability in order to achive a partitioning of the data space. [151]

The basic methodology for Gaussian Mixture Model (GMM) [105] works as follows. First, calculate the Gaussian mixture distribution with k components for a d-dimensional vector $x \in \mathbb{R}^d$. In other words, fit k probability distributions (Gaussians) to the data (see Equation (3.7)) where $0 \leq \pi_l \leq 1$.

$$p(x) = \sum_{l=1}^{k} \pi_l \times \mathcal{N}(x|\mu_l, \Sigma_l) \tag{3.7}$$

Then, recalculate the mean, find the centre of mass and, what differs from K-Means, calculate the covariance matrices. The most popular approach for doing this is the GMM using Expectation Maximization (EM) [105]. Its aim is to optimise the log-likelihood of points being generated by these k distributions. The two EM-steps are finished when the Maximum Likelihood Estimate (MLE) i.e. the parameters θ_{ML} with maximum likelihood are found, see Equation (3.8). In other words, MLE indicates which parameters are best to fit the model onto the data:

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}}\{p(X|\theta\} \tag{3.8}$$

where

$$p(x|\theta) = \prod_{i=1}^{n} p(x_i|\theta) \tag{3.9}$$

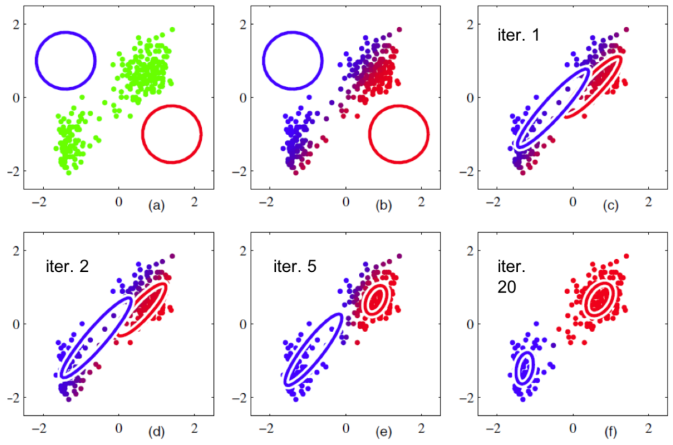


Figure 3.7: EM Probabilistic Clustering: The circles illustrate two random Gaussians to partition the data points. With each step, the mean and covariance matrices are re-calculated, where smaller the ellipses describe better clustering. [23]

In the calculation of the log-likelihood in the EM, there are mutual dependencies between the means μ and the weights γ. These depedencies can be broken by optimizing them independently in two steps, namely in the expectation and the maximization step. These operations are repeated iteratively until convergence is reached. The EM algorithm works as follows: [121]

1. Inizialize means μ_j, covariance Σ_j, and mixing coefficiets π_j, and evaluate the initial log-likelihood.

2. **E-Step**: Evaluate the responsibilities using the current parameter values:

$$\gamma_j^{new}(x_i) = \frac{\pi_j \times \mathcal{N}(x_i|\mu_j, \Sigma_j)}{\sum_{l=1}^{k} \pi_j \times \mathcal{N}(x_i|\mu_l, \Sigma_l)} \tag{3.10}$$

3. **M-Step**: Re-estimate the parameters using the current responsibilities:

$$\mu_j^{new} = \frac{\sum_{i=1}^{n} \gamma_j^{new}(x_i)x_i}{\sum_{i=1}^{n} \gamma_j^{new}(x_i)} \tag{3.11}$$

$$\Sigma_j^{new} = \frac{\sum_{i=1}^{n} \gamma_j^{new}(x_i)(x_i - \mu_j^{new})(x_i - \mu_j^{new})^T}{\sum_{i=1}^{n} \gamma_j^{new}(x_i)} \tag{3.12}$$

$$\pi^{new} = \frac{\sum_{i=1}^{n} \gamma_j^{new}(x_i)}{\sum_{l=1}^{k} \sum_{i=1}^{n} \gamma_l^{new}(x_i)} \tag{3.13}$$

4. Evaluate the new log-likelihood $\log p(X|\theta^{new})$ and check for convergence of parameters $|\log p(X|\theta^{new}) - \log p(X|\theta)| \leq \epsilon$. If the convergence criterion is not satisfied yet, set $\theta = \theta^{new}$ and go back to step 2.

In conclusion, probabilistic methods yield better results for data with varying size of clusters or clusters having different variances than a classical K-Means approach. For the sake of complexity, EM is however often combined with partitioning approaches to get the best of both algorithms. One of its major problems is the selection of a good number k: If the value is too high, the risk of overfitting is also increased. If the value is too low, the model may not fit the data at all. The value for k can be determined equivalently to the methods described in Section 3.3.1.

3.3.3 Density-Based Methods

Partitioning-based methods only work for convex data structures. The basic idea of density-based clusters is, that a data space does not only consist of a certain number of data point clusters. Rather it assumes that each data space contains a number of outliers, i.e. points in regions of lower density, that separate the existing clusters, i.e. points in dense data regions. The basic functionality of density-based methods will be explained by means of DBSCAN.

A partitioning $\{C_1, ...C_k, N\}$ of the database D consists of $C_1, ..., C_k$ density-based clusters and the noise $N = D \setminus (C_1 \cup ... \cup C_k)$. In order to determine the local point density at a certain point q, two parameters have to be defined. [121]

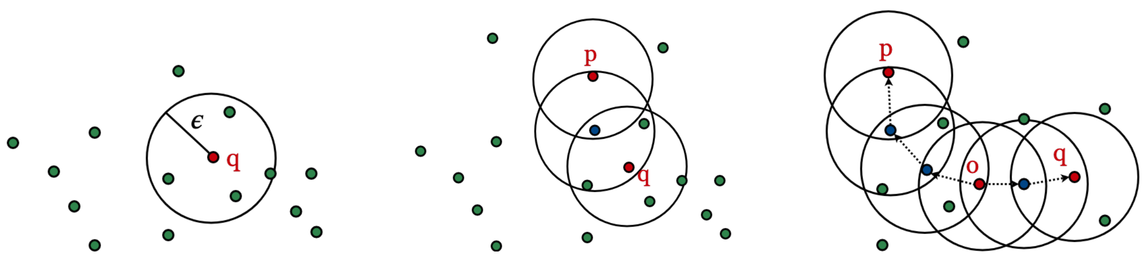
1. ϵ-radius for the neighbourhood of point q

$$N_\epsilon(q) = \{p \in D | dist(p, q) \leq \epsilon\} \tag{3.14}$$

2. *MinPts*, the minimum number of points in the given neighbourhood $N_\epsilon(q)$

The above parameters are determined as follows: First, fix a value for *MinPts*. The default is $2d-1$ where d is the dimension of the data space. Then compute the k-distance for all points $p \in D$ with $k = minPts$ and create a k-distance plot showing the k-distances of all objects in decreasing order. Lastly, select the border object o from the *MinPts*-distance plot and set ϵ to *MinPts*-distance(o). [43, 119, 121]

After selecting the parameters, a point q is taken as the core point of initialisation. A point p in the data space is directly density-reachable from q if $p \in N_\epsilon(q)$ and q is core object w.r.t. ϵ and *MinPts*. The major concepts of (directly) density-reachability and density-connectivity are explained in Figure 3.8, and a formal algorithm is provided in Algorithm 4. [121]



(a) Points that are directly density-reachable from the core point q.

(b) p is density-reachable, i.e. the transive closure of (a).

(c) p is density-connected to q where both points are density-reachable from a point o.

Figure 3.8: Concept of density-based algorithm: The basic theorem is that each object in a density-based cluster C is density-reachable from any of its core-objects while nothing else is density-reachable from core objects [43, 121].

```
forall $o \in \mathcal{D}$ do
    if $o$ is not yet classified then
        if $o$ is a core-object then
            Collect all objects density-reachable from $o$ and assign them to a
              new cluster;
        else
            Assign $o$ to noise $N$;
        end
    end
end
```

Algorithm 4: Algorithm for the **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise (DBSCAN) [121]

One of the major advantages of density-based methods is the flexibility of data in the n-dimensional space. In this space, clusters can have any shape and are not restricted to convex shapes. Also, the number of clusters is determined automatically, hence it is less prone to poor initialisation, than for instance K-Means. Further, these algorithms have the ability to separate clusters from surrounding noise, which can be inherently helpful compared to centroid-based approaches where noise can have a negative impact on the entire clustering result. [121]

However, DBSCAN and comparable algorithms also have drawbacks. For example, the result of the clustering highly depends on good input parameters. However, these parameters are difficult to determine and may therefore negatively influence the clustering quality. [121]

An alternative density-based algorithm for DBSCAN is the Mean Shift algorithm, where the iterative mode search is applied for each data point. Points that converge to the same mode, also referred to as *basin of attraction*, are subsequently grouped together. Mean Shift is computed following these steps: [121]

1. Select a window size ϵ, and a starting position m.
2. Calculate the mean of all points inside the window $W(m)$.
3. Shift the window to that position.
4. Repeat until convergence is reached.

A extension is the weighted Mean algorithm where different weights for the points in the window are calculated by some kernel k. A second variant is binning: Therefore, data points are quantised to a grid and the mode seeking iteratively is applied only once per bin.

A major drawback of Mean Shift is its relatively high complexity. On the positive side, the algorithm is not restricted to convex shapes, is robust to outliers, and easy to implement. Furthermore, it only has one single parameter, ϵ, and the number of clusters is determined automatically. [121]

3.3.4 Hierarchical Methods

As the name predicts, hierarchical clustering methods build a group of clusters that can be depicted as a hierarchy of clusters. The clusters are thereby presented hierarchically where the number of cluster depends on the height chosen in the dendrogram, see Figure 3.9.

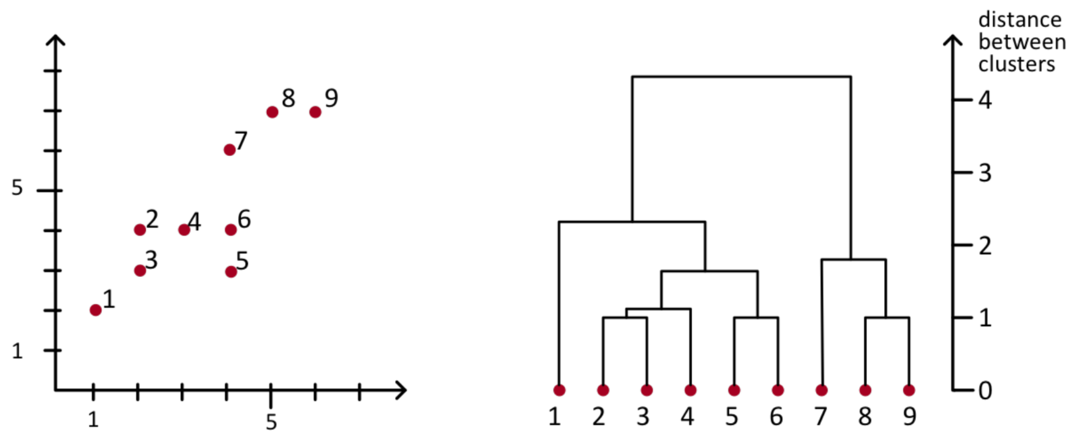


Figure 3.9: Hierarchical representation of document clustering using a dendrogram. [121]

There are two approaches to hierarchical clustering. Firstly, clusters can be built top-down or *divisive* where the complete document collection forms one cluster. Then the cluster is split recursively into sub-clusters. [8] At each step of the process, the dissimilarity between clusters is calculated in order to decide whether splitting is required or not [143]. Secondly, hierarchies can be build using the bottom-up or *agglomerative* approach, where initially each instance, i.e. each data point, constitutes its own cluster respectively, and then the closest clusters are merged together [8,143]. Hierarchical methods work similarly to partitioning methods with distance-based algorithms to measure the closeness between text documents. There are several approaches for measuring the similarity between all documents in the collection. An overview is given in Table 3.3.

Method	**Description**
single-linkage	calculates the minimum distance
complete-linkage	calculates the maximum distance
centroid-linkage	represent clusters by centroids not for pairwise similarity
average-linkage	calculate all pairwise distances to find average
group-average	calculate average pairs from the same original cluster
Ward's	calculate the increase in the sum of squares of the distance of instances from centroid, before and after fusing two clusters

Table 3.3: Strategies for merging the closest clusters in hierarchical clustering approaches, adapted from [143].

3.3.5 Kohonen Self-Organizing Maps

As opposed to other kinds of neural networks, SOMs consist only of one input layer and one output layer, while not including any hidden layers. The input layer contains the raw information, that is a collection of n-dimensional data vectors. The output layer is the SOM itself, also known as topological map, or feature map. It is usually represented two-dimensionally in a grid structure, however it can also be one-dimensional or three-dimensional [12, 67]. Every node in the feature map is associated with a dynamic weight vector that stores all the learning processing and knowledge and is hence the "memory" of the SOM. Although invisible to the user, the weight vector has the same structure as the data input vector, i.e. they have the same dimensionality, share the same attribute definitions and the same attribute sequence. Each neuron in the topological map is made up of a high-dimensional weight vector, the weight vectors however cannot determine the positions of their corresponding nodes in the visual space [153]. In that mapping, topological relations should be preserved, i.e. patterns that are close in the input space will be mapped to units that are close in the output space. Supposing that each SOM unit represents a cluster centre, a k-unit SOM will be able to perform similar tasks to a K-Means algorithm. An illustration is given in Figure 3.10.

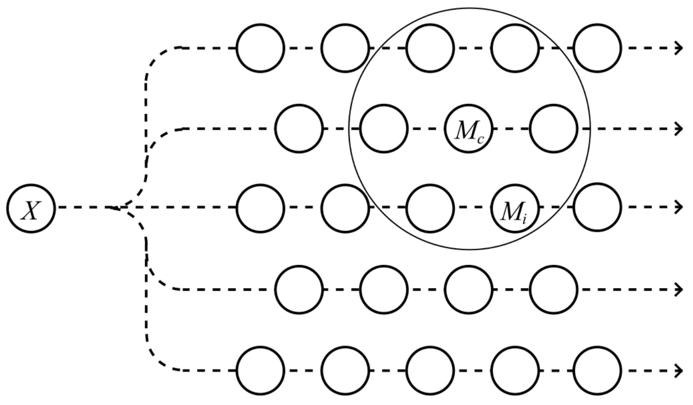


Figure 3.10: The hexagonal structure of a SOM. M_c shows the winning node that has impacts on its surrounding neurons (M_i) [67].

The iterative training processing works as follows: Firstly, every weight vector w is initialised, such that $0 < w < 1$. Then an input vector is randomly chosen and presented to the output feature map. In the output map, every node is examined to calculate the neuron whose weight vector is the most relevant to the input vector, i.e. calculating its smallest Euclidean distance. This resulting winning neuron is commonly known as the Best Matching Unit (BMU). Next, the radius of the neighbourhood of the BMU is calculated. The radius usually starts large such that it has a global impact on the output grid, and diminishes at each time step. The weights of each node inside the neighbouring area are adjusted with a high impact on the closest neighbouring nodes and decreasing impact on nodes farther away within the radius. This process is repeated iteratively until convergence is reached, which is when the neighbourhood area shrinks to the winning node itself. A formal algorithm for self-organizing and labelling the grid is given in Algorithm 5.

To help users understand cluster formations, it is inherently useful to label the document clusters, such that the system allows for searching documents regarding a certain topic. Therefore, nodes with similar weight vectors can be merged and labelled as one topic area. This can be done by the following trivial approach for each node: Locate the largest value in the weight vector and find its corresponding term in the data vector. Then, assign the term to the node as the winning term, and merge nodes sharing the same term as a region [153].

```
input  : Collection of raw data vectors
output: SOM
Initialization of the parameters, neighbourhood radius, and weight vectors in
 the feature map;
while the converge condition is not satisfied do
    1. Randomly pick up a raw data vector as an input vector;
    2. Calculate the winning node whose Eucidean
    distance is the smallest between the data vector
    and the weight vector associated with the node;
    for all neighbouring nodes of the winning node do
        update their weight vectors;
    end
end
foreach raw data vector in the collection do
    1. Find its winning node in the feature map;
    2. Assign it to the winning node;
end
foreach node in the feature map do
    1. Label the selected node;
    2. Merge adjacent nodes sharing the same term(s);
end
```

Algorithm 5: SOM algorithm adapted from [153].

3.3.6 Others

Due to the very limited time frame, not all clustering algorithms can be considered in the implementation part of this work. However, for the sake of completeness, two additional clustering methods will be presented in this section: incremental clustering and spectral clustering. Experiments regarding the effectiveness on our test data will be left for future work.

Incremental Clustering

As opposed to other methods where the document collection is treated all at once, incremental clustering works instance by instance. The clusters are thereby updated step by step appropriately, which can also lead to radical reconstruction during the processing of the next instance. The key process is to maximise the category utility, that is finding the host that produces the greatest category utility for the split at a certain level in the tree. The tree is used for clustering where all instances are represented by the leaves and the root comprises the entire dataset. [143]

Spectral Clustering

Spectral Clustering is a graph-based technique, that represents the data by a similarity graph and unveils the structural properties with the eigencomposition of an associated Laplacian matrix [129]. Basically, each vertex in the graph is represented by a vector of its corresponding components in the eigenvectors and K-Means applied to the first k eigenvectors.

In more detail, spectral clustering algorithms work as follows: Firstly, a similarity graph $G = (V, E)$ is constructed, where each vertex v_i represents one data point x_i. In case G is a weighted graph, edges in different clusters are weighted with low weights, while edges within a cluster are assigned higher weights. Two common weighted versions of a similarity graph that are regularly used in spectral clustering are the KNN graph and the fully connected graph. The first is a non-symmetric graph that connects a vertex v_i with v_j if v_j is among the k nearest neighbours of v_i. The latter simply connects all points with positive similarity and weighs the edges according to their similarity. Note that the graph does not have to be weighted, as in the ϵ-neighbourhood graph where all points are connected whose pairwise distance is smaller than a certain threshold ϵ [137].

After constructing the graph, its Laplacian L, i.e. the undirected version of a directed graph, is computed along with the first k eigenvectors $u_1, ..., u_k$ of L that are ranked in decreasing order.[1] [90] The eigenvectors are presented as a matrix $U \in \mathbb{R}^{n \times k}$ where each row in the matrix $i, ..., n$ is a vector $y_i \in \mathbb{R}^k$ that is clustered with a classical algorithm such as K-Means. Lastly, the output clusters $C_1, ..., C_k$ are remapped onto a new dimensional space yielding the spectral clustering outcome [129]. Opposed to various other algorithms, the clusters may have arbitrary shape and the algorithm can be implemented in nearly linear time [28, 71].

3.4 K-Nearest Neighbour Search

Another possibility to boost performance of WiTTSim is to integrate a KNN search for determining the k nearest neighbours of given a point p. The least intuitive approach simply is a brute-force computation that performs a pairwise search on all datapoints. For N samples in d dimensions this leads to a complexity of $O(dN^2)$ [53]. However this is only effective on small datasets. For circumventing the costly process of comparing all datapoints to each other – especially for larger data samples – a number of approaches exist based on KNN that involve structuring the data beforehand. This can be achieved

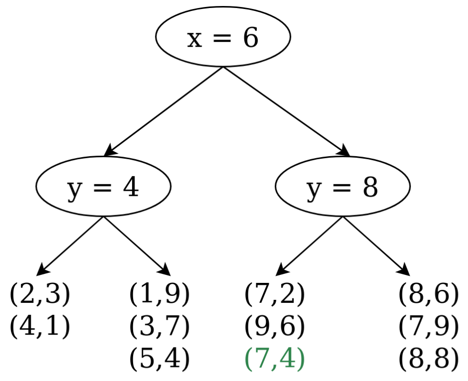
[1]An overview of calculating the eigenvectors in different ways is given in [141].

by, for example, using a k-d tree, or creating a balltree of the data. Both algorithms order the data effectively in advance and will be described in the following subsections.

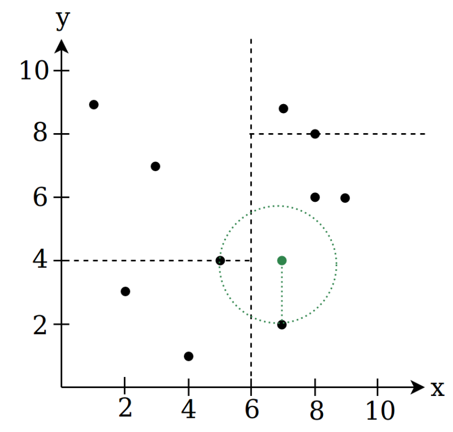
3.4.1 K-d tree

The k-dimensional k-d tree[2] [49] is a multidimensional binary search tree, that is, a data structure for the storage of k-dimensional data. The k-d tree is structured as follows: The root in the search tree represents the entire dataset, while each further nonterminal node is a successor node, representing the subfiles. The tree leaves store the records, called buckets. In k dimensions a record is represented by k keys [49].

The basic algorithm to create a k-d tree works as follows: Pick a random dimension x, find the partition value, which is the median value of all data points in that dimension, and split the data at x. Values that are smaller than the median belong to the left son node, while values equal to or bigger than the median will belong to the right son. These steps are repeated iteratively with the next random dimension until all dimensions are split [49]. An illustration of the algorithm with an example dataset of ten points in two dimensions is given in Figure 3.11.



(a) K-d tree for a two-dimensional dataset.



(b) Datapoints depicted in the two-dimensional space. Dashed lines indicate the tree branches.

Figure 3.11: Creating a k-d tree structure from a two-dimensional dataset containing ten random data points: Points that are located on the left branch are smaller than the root value, while points located to the right are equally high or higher than the root. The green input point (7,4) is located in the same two-dimensional space, and integrated in the search tree accordingly.

Due to the efficient ordering of the tree structure, only the data points closest to the query point will have to be examined. In more detail, the search algorithm works as follows: Firstly, the root node is examined. If the node under investigation is not a leaf node, the node on the same side of the partition as the query record is called recursively. If the node is a terminal node, all data points in the bucket are examined. After calculating

[2]The original k-d tree was presented in 1975 by Bentley [17]. For retrieving the m closest matches, they adapted their algorithm in 1977 [49]. Since the algorithm in this work will be used solely for the KNN search, only the latter so-called *optimized k-d tree* will be regarded and be referred to as *k-d tree.*

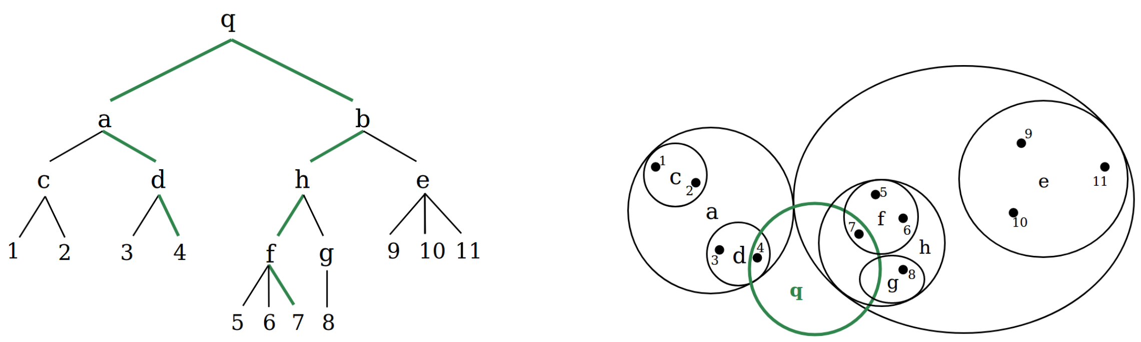
the distances to all points in the bucket, a list of m closest data points is created and maintained during the whole procedure. In case a closer point is found during the tree search, the priority list will be updated.

In some scenarios however, points in other buckets will be as close as or even closer to the query point (see point (5,4) in Figure 3.12). Therefore, subsequent to examining the respective bucket, a control is performed that decides whether the opposite side of the partition has to be taken into account additionally. This test – also referred to as *bounds-overlap-ball* test – draws a circle around the query with a radius equal to the distance to the closest record encountered so far. If the test fails, none of the datapoints of the neighbouring bucket belong to the m closest data points to the query point. Conversely, the test is effective when the radius intersects with a border of another non-termining node. In that scenario, a *ball-within-bound* test is performed that examines if any data points lie withing the radius. If the test retrieves any data points within the radius, the intersecting bucket will have to be considered as well. If it fails, the current list of m records is correct and no further datapoints have to be added or replaced.

3.4.2 Balltree

Although the balltree algorithm is based on the k-d tree algorithm, sibling regions in balltrees are allowed to intersect [92]. A balltree is a complete binary tree in which each ball, that is the region bounded by a hypersphere in the n-dimensional Euclidean space $\mathbb{R}^n$, contains the balls of its children. The most inner balls, i.e. the leaf nodes, hold the data points themselves.

The top-down algorithm works recursively: Firstly, the dimension with the highest variance in the data is chosen, and the median value in that direction determined. Again, the root node holds the entire dataset. At each stage the ball is split into two parts, becoming the left and right children of the root node. Whether the split ends on the left or right side depends on their coordinate value in the given dimension. If the value is smaller than the median value, the data split lands in the left successor, in all other cases it is grouped with the right successor. These steps are repeated recursively until all data points form leaf nodes. [92] An example of a binary balltree and its corresponding balls in the plane is illustrated in Figure 3.12.



(a) Balltree for a two-dimensional dataset. (b) Balls depicted in two-dimensional space.

Figure 3.12: Creating a balltree. The data is separated by means of ellipsoidal hyperspheres in the data space, creating the tree with a top-down algorithm. Illustration adapted from [39].

3.5 Evaluation Metrics

A large number of different evaluation techniques exist in the field of clustering, which can be grouped into three major categories: Supervised methods rely on available ground truth labels for the entire dataset in order to assess the quality of the clustering. In natural settings however, these labels are often not available, because if they were, clustering the data would be redundant. Conversely, unsupervised algorithms do not rely on any labels, but the quality of the clustering is only captured externally without any internal information of the data known. A mixture of both types are so-called semi-supervised methods, where only a part of the data is manually labelled and the conclusions drawn on the entire dataset. All types of methods have advantages and drawbacks and will be presented in further detail in the following subsections.[3]

3.5.1 Unsupervised Methods

One of the most popular techniques for measuring clustering quality externally is calculating its silhouette coefficient which states how accurate a given cluster is [111, 126]. That is, the coefficient denotes how well the clusters are separated from each other and can further reveal whether points are assigned to the wrong cluster. In more detail, the silhouette coefficient measures the average distance between a point p and all its cluster neighbours. Thereafter, the distance between p and its next cluster is computed. Optimal results are achieved when the distance to its own centre is minimal while the distance to the next cluster centre is maximal. By noting $a(p)$ the mean distance between p and all other points in the same cluster and $b(p)$ the mean distance between p and all other points in the next nearest cluster, the silhouette coefficient $s(p)$ for a single sample p is defined as follows [70]:

$$s(p) = \frac{b(p) - a(p)}{max(a(p), b(p))} \tag{3.15}$$

This yields a result between -1 and 1 where 1 is the perfect classification and -1 fails to assign the clusters correctly. The silhouette coefficient for all data points is computed by the average value of all values $s(p)$:

$$S(p) = \frac{1}{|P|} \sum_{p \in P} s(p) \tag{3.16}$$

where the total result also lies within the span of -1 and 1.

Another measure to determine the cluster accuracy is determining the *Calinski-Harabasz index* [26]. By noting k the number of clusters, B_k and W_k the between and within-clustering dispersion matrices respectively, the Calinski-Harabasz index $s(k)$ indicates how well a clustering model defines its clusters, such that the higher the score, the more dense and well separated the clusters. It is defined as

$$s(k) = \frac{T_r(B_k)}{T_r(W_k)} \times \frac{N - k}{k - 1} \tag{3.17}$$

[3]However, due to the large amount of evaluation techniques it is not possible to give an exhaustive list of all methods. For a larger overview of evaluation methods please refer to Wagner and Wagner [138].

where

$$B_k = \sum_{j=1}^{k} n_{c_i}(\mu_{c_i} - \mu)^T \qquad W_k = \sum_{i=1}^{m}(x_i - \mu_{c_i})(x_i - \mu_{c_i})^T \tag{3.18}$$

As opposed to the two metrics presented above, the Davies-Bouldin index [37] defines the lowest possible score, that is zero, to be the best cluster separation, while higher results indicate poorer defined clusters. The index defines the average similarity between each cluster C_i for $i = 1, ..., k$ clusters and its most similar one C_j. Similarity is defined by R_{ij}:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \tag{3.19}$$

where s_i is the average distance between each point in C_i and d_{ij} is the distance between cluster centroids i and j. The Davies-Bouldin index is then defined as follows [96]:

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j} R_{ij} \tag{3.20}$$

3.5.2 Supervised Methods

However, unsupervised evaluation criteria presented in the previous subsection, often only evaluate the clustering on a very theoretical background. In order to evaluate the clustering results based on their internal structure, a so-called "ground truth" should be determined for a certain subpart of the data and then the corresponding clustering algorithms compared using precision and recall. The confusion matrix in Table 3.4, shows the comparison values between the actual class and the class predicted by the system.

		Predicted Class	
		+	–
Actual Class	+	**TP** True Positives	**FN** False Negatives Type II error
	–	**FP** False Positives Type I error	**TN** True Negatives

Table 3.4: Fourfold table reflecting how the values TP, FN, FP, and TN are determined by comparing the actual class to the predicted class.

To make the clustering results comparable, the established measures precision and recall are determined. While the recall value reflects what proportion in the actual class was labelled correctly, precision defines what proportion of the output is correct. Equation (3.21) shows how the values in Table 3.4 are utilised to calulate precision and recall.

$$\text{Recall (R)} = \frac{\text{TP}}{\text{TP+FN}} \qquad \text{Precision (P)} = \frac{\text{TP}}{\text{TP+FP}} \tag{3.21}$$

Both values oppose each other, which is why they should not be treated separately but better be combined using F-measure. The coefficient β is used to either favour precision or recall. For calculating F_1, i.e. $\beta = 1$, both measures are treated equally.

$$\text{F-measure} = \frac{(\beta^2+1)\mathrm{PR}}{\beta^2\mathrm{P{+}R}} \Rightarrow \mathrm{F}_1 = \frac{2\mathrm{PR}}{\mathrm{P{+}R}} \tag{3.22}$$

Other methods based on the fourfold table are for instance the Rand Index [103] which is defined as

$$\mathrm{RI} = \frac{TP+TN}{TP+FP+FN+TN} \tag{3.23}$$

and the Adjusted Rand Index (ARI) [63,136,139].[4]

Another evaluation method based on ground truth labels is V-measure, an external entropy-based cluster evaluation measure that combines homogeneity and completeness. These two criteria capture a clustering solution's success by including all and only data points from a given class in a given cluster. Homogeneity states that each cluster contains only members of a single class, while completeness reflects whether all members of a given class are assigned to the same cluster [109].

Lastly, approaches that are based on MI comprise Normalized Mutual Information (NMI) [84,125] and Adjusted Mutual Information (AMI) [136]. NMI allows for measuring and comparing results between different clusterings having different number of clusters, while AMI is a correction of MI that adjusts the score accordingly when the number of data points is relatively small compared with the number of clusters.[5]

3.6 Summary

This section summarises the chapter briefly by revising its most essential methods. Before clustering the high-dimensional dataspace, dimensionality reduction techniques are reviewed. Several approaches exist. The most common unsupervised linear reduction techniques comprise SVD, PCA, and SRP. SVD is based on the eigenvalue decomposition calculating singular values, while PCA computes the covariance matrix of a dataset to determine a lower dimensional space that maintains as much variance of the original data space as possible. SRP is especially developed for sparse datasets, projecting the data onto a random lower dimensional space. A supervised version of linear reduction techniques is the Linear Discriminant Analysis (LDA), where the separability among known categories is maximised. Manifold reduction techniques include t-SNE and UMAP. The latter incorporates ideas from topological data analysis and thereby constructs a weighted k-neighbour graph.

Various clustering methods exist in the field of document clustering. Two major distinctions can be made, namely flat clustering and hierarchical clustering. Flat clustering is then again divided into partitioning clustering, density-based clustering, and probabilisitic clustering. Partitioning algorithms are for example K-Means or C-Means clustering, where the latter is a soft clustering technique that allows data points to belong to multiple classes. Both methods split the data based on random centroids and assign each data point to its closest centroid. The centroids are then recalculated until the dataset converges. Density-based clustering methods base on the assumption that dense data regions should be grouped together, and no cluster centre has to be defined. Examples

[4]For a detailed definition please refer to Santos and Embrechts [116].
[5]For a detailed definition please refer to Learned-Miller [74].

include DBSCAN and Mean-Shift clustering. Probabilistic models assign probabilites to data points, resulting in a soft clustering, where each point belongs to a cluster with a certain probability. An example for this is the GMM. As for the hierarchical clustering methods, one distinguishes divisive, i.e. top-down, and agglomerative, i.e. bottom-up approaches. In divisive approaches, the entire document collection forms a cluster and is then split recursively into subclusters, whereas in agglomerative algorithms each data point forms its own cluster and closest clusters are merged together iteratively.

Apart from incorporating document clustering in the similarity search, a structured KNN search can also be incorporated. Therefore, the datapoints are not only compared pairwise but structured trees are built from the entire dataset and then only respective branches searched that match the input query. Two well-known methods are the k-d tree search and the balltree algorithm. While the former separates the data linearly in each dimension, the latter allows for separating the data with ellipsoidal hyperspheres in the data space.

Lastly, evaluation metrics have been presented, namely supervised and unsupervised methods. Supervised methods rely on the ground truth labels available, however labelling large amounts of data is immensely time-consuming and costly. Also, retrieving the labels is in most cases the goal and original purpose of clustering, which is why they are mostly not available in natural tasks. In other words, with labels available, the task of clustering becomes fairly redundant. In that case, unsupervised methods allow for an evaluation without any ground truth provided. However, results should be treated with caution since only external parameters are considered, and the internal structure of the data is not taken into account.

4 Implementation

The implementation of the relevant methods from Chapter 3 will be explained in the following sections. This includes the data preprocessing to be carried out prior to the clustering task itself (Section 4.2) and the feature selection or dimensionality reduction that is necessary for an efficient search process (Section 4.3). In addition, the implementation of the experiments will be shown (Section 4.4) and the final integration of the presented clustering into WiTTSim will be explained in further detail.

4.1 Data Collection

The underlying data for the dimensionality reduction and document clustering are the literary remains of the philosopher Ludwig Wittgenstein, his *Nachlass*, which has been provided by the Wittgenstein Archive Bergen (WAB) in cooperation with the Center for Information and Speech Processing (CIS). The Nachlass comprises approximately 18,000 pages. Most of them were unpublished after Wittgenstein's death, now however they are openly accessible[1] [99, p. 4]. The corpus has been subdivided into 55,000 logical paragraphs, so-called *remarks*, that form the database of this work. Each remark is labelled with a unique identifier which is its siglum.

The data, meaning all the remarks, has already been tagged and underlies the XML structure that encodes various information about the original manuscript or typescript, respectively, such as the date of writing, deletions, insertions, and alternatives. An exemplary file is shown in Source Code 4.1.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<body>
  <ab n="Ts-230b,11[3]" ana="abnr:44" date_norm="1945-08-01?-1945-08-31?">
    <w t="PIS" l="man">Man</w>
    <w t="VMFIN" l="können">könnte</w>
    <w t="PRF" l="sich">sich</w>
    <w t="NN" l="Mensch">Menschen</w>
    <w t="VVINF" l="denken">denken</w>
    ...
  </ab>
  <ab n="Ts-230c,11[3]" ana="abnr:44" date_norm="1945-08-01?-1945-08-31?">
    ...
  </ab>
</body>
```

Source Code 4.1: XML structure required for generating the feature vectors. The required information contains the anonymous blocks (ab) that contain the siglum, and the words (w) containing the tags (t) and lemmas (l) that are required for calculating the dictionaries and subsequently the vectors.

[1]Transcriptions accessible at http://wab.uib.no/transform/wab.php?modus=opsjoner

The preprocessing has already been implemented in previous work in WiTTSim [132] and has been improved by Tan et al. [127] later for a more efficient handling of XML formatting instead of processing plain text files. The preprocessing includes punctuation removal, tokenization, lemmatization, stop word filtering, and POS-tagging. Every remark is treated as a single document for the clustering. For training purposes, the data has been split to smaller sets which enables a more accurate evaluation. Also, for a small subpart of the remarks, ground truth labels are available.

4.2 Data Preprocessing

Prior to clustering the data, some preprocessing steps are carried out. This is because WiTTSim demands a representation in form of vectors containing certain features, such as synonymous words or POS-tags. For a more detailed description of the feature set please refer to Section 3.1. Since there are multiple different setups for the experiments, several files will be created, one for each setup. The files contain dictionaries where each key stores the source information, i.e. the siglum of each remark, and each value stores the vector itself in the format *siglum:vector* in the binary files. The binarisation has been implemented in previous work using the python library *Pickle*[2] and is used again in this work for retrieving the input data.

For building the binary files, several preprocessing steps need to be carried out. Firstly, the datafile path for all normalised remarks has to be specified. All required remarks for one dataset are specified in an Excel spreadsheet, along with their ground truth labels if they are available. Then, the remarks in the Excel are extracted from the entire Nachlass and saved to an output folder.

After extracting the remarks in the required formatting, some of the words still contain XML tags, such as separated words containing a newline tag. For all those words, some further manipulation is necessary, such as simple post-processing methods that remove newline tags:

```
<w t="NN" l="Schweigen">Schwei&lt;lb rend="shyphen"/&gt;gen</w>
```

```
<w t="NN" l="Schweigen">Schweigen</w>
```

Once the clean XML files are created, the megavectors can be extracted. This is done in WiTTSim by creating the dictionaries for all features, translating these dictionaries into vectors and saving them into a binary file. The binary file stores all remarks as keys in a dictionary structure along with their megavectors as values. Note that the megavector size varies from dataset to dataset. As for all 54,930 remarks, the megavector size comprises 115,601 elements long with 45,337 words, 22,739 lemmas, 38 POS-tags, and 47,487 synonyms.

[2]https://docs.python.org/3/library/pickle.html

4.3 Dimensionality Reduction

At the beginning of the experiments, the dimension of the feature space is relatively high, depending on the size and complexity of the dataset. In other words, by incorporating remarks with a vocabulary that is rich in variety, more vector positions will be necessary in order to map the higher number of words and synonyms to positions in the vector than, say, a very short text with only few different words. Evidently, the largest dimension in the Wittgenstein Nachlass will be attained when processing the entire document collection: with regard to all features, the megavector for each remark then stores 115,601 features.

Clustering datapoints in a 115,601-dimensional space is not only tremendously inefficient but will also fail to retrieve appropriate clusters due to the data sparsity in high dimensions. Therefore, the dimensions will be reduced beforehand to a more reasonable amount of features. However, different reduction techniques will yield different results and not all methods will be equally applicable to our datasets. For determining a suitable algorithm, four different dimensionality reductions will be tested and briefly evaluated. The resulting reduced data will be utilised for the following clustering approaches.

Please note that not all of the methods presented in Section 3.2 will be tested. This is on the one side due to time restrictions of this work, and on the other side due to the limited access to ground truth labels that are essential for any supervised dimensionality reduction method. In the experiments, only very few ground truth labels are available while a vast majority of roughly 99% of the data is unlabelled. Hence semi-supervised or entirely supervised methods such as LDA are not suitable for the experiments and will not be considered further in the comparison of reduction approaches. Instead, four unsupervised methods will be examined, namely PCA, SVD, SRP, and UMAP. The dimensionality reduction methods will be implemented using scikit-learn[3] [96], a SciPy toolkit for machine learning implementations and the python package UMAP[4] [86], for the UMAP dimensionality reduction.

PCA is a linear dimensionality reduction algorithm using SVD to project the data to a lower dimensional space. The optimal number of dimensions for PCA is determined as depicted in Section 3.2.2. Ideally, after the reduction with the optimal number of dimensions, still 80-90% of the variance is kept [71]. Since PCA can be tricky for sparse feature spaces, SVD for high-dimensional sparse feature spaces will be compared additionally. The ideal number of dimensions can be found by visualising the variance in a chart, also known as *scatter*. When the scatter shows a step in the data, the optimal number of dimensions is reached. In case no step can be taken from the illustration, ideally a variance of 80-90% should be kept. Especially for the large feature vectors of the entire Nachlass, SRP can be useful and should therefore be tested and evaluated furthermore. Moreover, UMAP will be taken into account, comparing the linear techniques above to a manifold technique for general non-linear dimension reduction.

Beside the methods listed above, another possibility is to apply hierarchical clustering on the feature space for dimensionality reduction to group together features that behave similarly, also known as feature agglomeration [71]. Due to limited evaluation possibilities in terms of determining an appropriate number of features, this method is skipped in the experiments.

[3]https://scikit-learn.org/stable/modules/decomposition.html#decompositions
[4]https://umap-learn.readthedocs.io/en/latest/index.html

4.4 Experimental Setup

Because different clustering algorithms can lead to entirely different results, it is important to run the algorithms on small subparts of the dataset in a (semi-)supervised way such that the clustering quality can be measured. It is crucial to emphasise that there is no perfect algorithm that can be applied to any kind of data. Hence the best algorithm for the underlying data, the Wittgenstein corpus, is still to be determined. In order to define the most suitable clustering algorithm, five experiments are carried out on several subsets of the dataset. Subsequently, the optimal algorithms should be applied to the entire Nachlass.

First, a toy corpus has been created to be able to evaluate the dimensionality reduction and clustering results in a supervised way. The corpus consists of 20 remarks that can be grouped into three clusters. Their semantic connectivity has been presented by Biesenbach [21]. For sake of the experiments, each of those remarks has been adapted in three different ways. Variant A substitutes words with their synonyms, variant B deletes words or sentences, and variant C permutes the sentences within the remark. This leads to a total of 80 remarks in the toy corpus with three clusters. A comprehensive and detailed list of utilised documents can be found in the appendices, Table A.1.

The data points in the toy corpus are distributed artificially, that is, the data groups are very clearly separated from one another. In order to avoid overfitting, more natural smaller datasets should be considered for the comparison and evaluation of the clustering algorithms. Therefore, the main works of Ludwig Wittgenstein have been included, namely the Prototractatus, the Big Typescript, the Philosophical Investigations, the Brown Book, and a test set of Wittgensteins Kringelbuch [110]. Each of the datasets is of different size and varying complexity, that is feature size, such that the dimensionality reduction methods and clustering algorithms can be evaluated on a higher variety of databases. All datasets are presented in further detail in Table 4.1.

For determining the most suitable clustering algorithm for each dataset, several different clustering algorithms are taken into account. Apparently, due to the large amount of existing clustering algorithms, not every algorithm can be tested on the datesets. However, algorithms of each category have been tested: K-Means for representing the partitioning algorithms, Mean-Shift and DBSCAN for density-based clustering, agglomerative clustering as a hierarchical clustering algorithm, and the GMM representing the probabilistic clustering approaches.

4.5 Integration into WiTTSim

Currently, the WiTTSim pipeline works as follows. The entire Nachlass is read in, transformed into feature vectors, and saved to disk. For each query, the query text is also transformed into a vector with the same features and dimensions as the precalculated dataset. Then the query vector is compared pairwise with each of the pre-saved 54,930 vectors. For retrieving the top k similar remarks, more than 30 minutes of computation time are required due to the immensely large number of features.

The dimensionality reduction and clustering can help to decrease the time drastically. That means, the results from previous sections are stored such that they can help boosting performance of WiTTSim and making it integrable into the WAST. An overview of the entire new pipline including the clustering beforhand is given in Figure 4.1 on page 40 and

Test set	Remarks	Dimensions	Samples	Clusters
Toy Corpus	Selected remarks of Ms-109[a], Ms-124, Ms-129, Ms-165, Ts-211[b], Ts-212, Ts-213, Ts-227a,b, Ts-228, Ts-230a,b,c	250	80	3
Prototractatus	Ms-104	6,618	881	unknown
Big Typescript	Ts-213	28,130	2,763	unknown
Philosophical Investigations	Ts-227a,b	18,481	1,434	unknown
Brown Book	Ts-310	8,832	325	unknown
Kringelbuch	Selected remarks of Ms-107, Ms-108	555	17	2
Full Nachlass	Ms-101 - Ts-310 (all)	115,601	54,904	unknown

[a] Manuscript
[b] Typescript

Table 4.1: Test sets used in the experiments along with their properties, i.e. what manuscrips or typescripts are included, their number of dimensions in the feature space, the number of samples, i.e. remarks, and their number of clusters if given.

the steps described in further detail in the following paragraphs.

In the first place, the dimensionality reduction model is applied to the dataset and saved to disk. This step is crucial in order to apply the same dimensions onto the query vector later. Next, the entire dataset is clustered and the clustering is saved beforehand into a dictionary (Source Code 4.2), where the key is the cluster ID and the value is a list of all the remarks that belong together. Note that the cluster ID is indispensable in order to map the cluster centroids to the clusters themselves containing the remarks later.

```
clusters = {
        Cluster_0:['Ts-230a,11[3]', ..., 'Ts-230c,11[3]'],
        Cluster_1:['Ts-211,398[2]', ..., 'Ms-109,224[3]'],
        Cluster_2:['Ts227a,243[3]', ..., 'Ts-228,71[3]' ]
        }
```

Source Code 4.2: Presaved Clusters in dictionaries.

When a user starts a query, the query is then projected onto the same dimensional space as the presaved datapoints. This is done by fitting the input vector to the same model and its parameters used for the entire dataset that has been saved in the previous step. Then the query input vector is transformed.

Instead of comparing the resulting vector to the entire data points, leading to an exhaustive search of the dataspace, only the distances from the vector to each of the cluster's centres is calculated (Source Code 4.3). Subsequently, the centroid with the smallest distance is returned, representing the cluster with the highest relevance (Equation (4.1)).

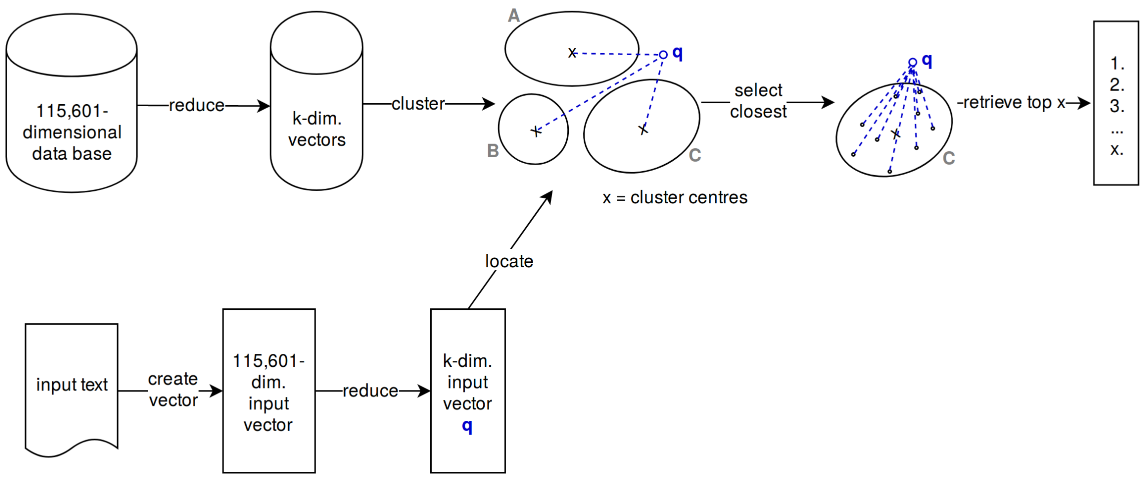


Figure 4.1: Integration of the dimensionality reduction and text clustering as preprocessing steps in the similarity search. The pipeline shows the feature reduction before clustering and the distance calculation to the centroids instead of the entire database. Then the respective cluster is retrieved and the top x similar remarks are retrieved.

Finally, the query is only compared to the remarks in the resulting cluster and an exhaustive search of the entire search space can be avoided. Again, the algorithm retrieves the points with the smallest distance to the query vector (Equation (4.2)).

```
centroids = {
        centroid_0:array([ 4.74749527e+00, -1.88144787e-01, -9.12357693e-01,
        1.81894323e+00, 3.04414634e+00,  3.05437675e-01,  1.76643242e-01,
        ... -5.27249285e-48]),
        ...
        centroid_k:array([ 5.25351076e+00, -7.28835828e-02,  3.82454004e+00,
        -2.64636965e-01, -8.37106511e-02, -1.23938670e-02, -2.84600658e-02,
        ... 7.14589766e-47])
    }
```

Source Code 4.3: Centroids saved in dictionary structure.

$$\text{determine\_cluster} = \underset{x \in \text{centroids}}{\arg\min} \, (dist(q, x)) \tag{4.1}$$

$$\text{closest} = \underset{y \in \text{clusters}}{\arg\min} \, (dist(q, y)) \tag{4.2}$$

In case the best fitting algorithm does not imply a centroid-based approach, the closest clusters cannot be determined. Therefore, KNN will be applied onto the reduced dataset for the determination of the n closest data points. The clustering algorithm is in that case solely used for data exploration and knowledge discovery and treated separately to the similarity search. Moreover, KNN can be deployed for determining the closest remarks for a data point in the database.

4.6 Evaluation

For the sake of completeness, both unsupervised and supervised methods have been presented in the methodological overview of this work in Section 3.5. However in consequence of the scarce ground truth available, that means there exists few if any ground truth labels, not all types of evaluation metrics are eligible for evaluating the clustering results. Before going into more detail, the available data will be presented such that the following algorithms can be suited onto the present situation.

The basis for the evaluation form solely the properties of the clustering outcomes themselves, which will allow for applying the following unsupervised metrics:

- Silhouette Coefficient
- Davies-Bouldin index
- Calinski-Harabasz index

Additionally, a list of 471 remarks that are each similar to up to 15 other remarks is given. The total number of likewise "labelled" texts comprise 1,670 remarks. Since this is accounts only for 3% of the total document collection, it would be naive to say that solely up to 15 remarks are allowed in one cluster. Hence, other remarks that are placed in the same cluster should be not punished by the evaluation algorithm. However, based on classical algorithms such as the fourfold table, a lot of false positives would be classified which would lead to poor results. Therefore, the only possibility is to reward clusterings that achieve to place correct remarks together in clusters, and to punish clusterings that fail to locate these "labelled" documents in the same buckets.

For incorporating the little ground truth available, the recall will be calculated for each of the available labels and the average value taken. The value between 0 and 1 will then indicate whether the known 1,670 remarks are correctly grouped, where 0 is the worst and 1 is the best clustering. Note that a calculation of precision and F-measure is elided due to reasons listed above.

5 Experimental Results and Evaluation

The following chapter will present the experimental results of the introduced methodologies in Chapter 4. The experimental results will be presented and evaluated in detail.

The chapter is further subdivided as follows. Section 5.1 will present the optimal feature space of the Wittgenstein corpus. Section 5.2 finds the best suitable clustering algorithm with the optimal number of clusters respectively. Next, the performance will be evaluated in Section 5.3. Furthermore, KNN results will be presented by comparing different parameters. Section 5.4 will explain arisen challenges and problems. The chapter concludes in Section 5.5, discussing the results and methods.

5.1 Optimal Feature Space

Obviously, the number of essential dimensionalities strongly depends on the original dataset and its respective number of features. This means that there is no perfect number of dimensions that fits on all presented datasets. Therefore, the optimal number of dimensions will be determined in dependence of each dataset. The critical point showing the optimal number of dimension is illustrated by a "knee" in the data graph. In case no breaking point can be detected, a maintained variance of about 80% should be strived for. Thereby a high spread in the data can be kept and the clustering algorithms can be applied onto the data properly. Note that this number highly varies and tends to increase as the datasets become more natural and the clusters are not created artificially. This can be explained through the fact that in most natural settings the data points will be separated less clearly from each other. This results in a larger number of features that have to be kept in order to guarantee a high variance overall.

Since not all dimensionality reduction methods are suitable for large and sparse datasets, an evaluation of the tested algorithms is given. The algorithms tested were PCA, SVD, SRP, and UMAP. PCA was however not applicable to our datasets because it allows only for reducing to a number of dimensionalities that is smaller than the number of samples. That means any number higher than the number of samples could not be tested.[1] For example, the Brown Book comprises 325 samples and 8,832 dimensions. Any dimension higher 325 could not be examined using PCA. However, for a small number of dimensions, PCA and SVD yield the exact same results, which is why the final results have been computed using SVD. All outcomes using SVD as the dimensionality reduction technique on all datasets are documented in Table 5.1 and will be explained later in further detail. A visualisation of the data graphs is presented in the appendices in Figures B.1 to B.3.

The subsequent method tested is SRP. Visualised in a two-dimensional space it seems that it fails to project the data onto a lower dimension. However, the 2D projection does not necessarily capture the most important dimensionalities. Performing SRP on the dataset however, yields slightly poorer results on the dataset than the results obtained using SVD. Detailed results are provided in Table D.1 on page 77.

[1]For a recapitulation of samples and dimensions, please refer again to Table 4.1 on page 39.

Next, UMAP is applied to the entire dataset. Because it is not a linear reduction technique, no variance calculation is possible. However, UMAP allows only for reducing the dataset to any dimension between 2 and 100, which is why the largest number is chosen in order to maintain as much information as possible. Already by reducing the data to 100 dimensions, the results look promising, as shown in Table D.2 on page 78. Being more sophisticated than linear models, it is however not possible to save the model because it is too large to be dumped into a file. Therefore the method is discarded, because without storing the model, it is impossible to reload it to apply it to further query data.

For these reasons, SVD is selected to be the best fitting algorithm. The major reasons were the promising results in combination with the clustering techniques and the simplicity of saving and restoring the model. Table 5.1 shows that for all underlying databases, only 3.06% to 10.28% of the features have to be maintained in order to keep 99% of variance. As opposed to the smaller datasets, a small loss of information for the entire Nachlass can not be avoided due to the large number of features. However, choosing the number of dimensions $d = 1,000$, already a high variance is kept, enabling a clustering on a reasonable number of features. Figure 5.1 shows that by keeping choosing $d = 1,600$, 80% of the original variance can be maintained.

Testset	**Original**	**Reduced**[a]	**Maintained**[b]
Toy Corpus	391	36	9.21%
Prototractatus	6,618	680	10.28%
Big Typescript	28,130	2,050	7.29%
Philosophical Investigations	18,481	700	3.79%
Brown Book	8,832	300	3.40%
Kringelbuch	555	17	3.06%
Full Nachlass	115,601	1,600[c]	1.38%

[a] reduced dimension such that 99% of variance is kept
[b] maintained proportion after dimensionality reduction
[c] for 80% variance

Table 5.1: Compared algorithms for dimensionality reduction and their results. All dimensions are calculated using SVD and are presented in relation to their original number of features.

5.2 Optimal Algorithm

Since there is no perfect algorithm that separates any dataset into perfect clusters, several clustering algorithms will be compared. For evaluation and performance reasons, the parameters will be determined first on the smaller datasets before clustering the entire Wittgenstein Nachlass of approximately 55,000 data points. However, since the dimensions and the type of data vary, multiple outcomes are likely. That is, well-separated datasets may be perfectly clustered by one algorithm while the same algorithm fails to separate more dense data regions. The experiments will comprise clustering techniques, as well as KNN approaches. A detailed comparison of the utilised algorithms on the presented dataset collection will be given in the following sections.

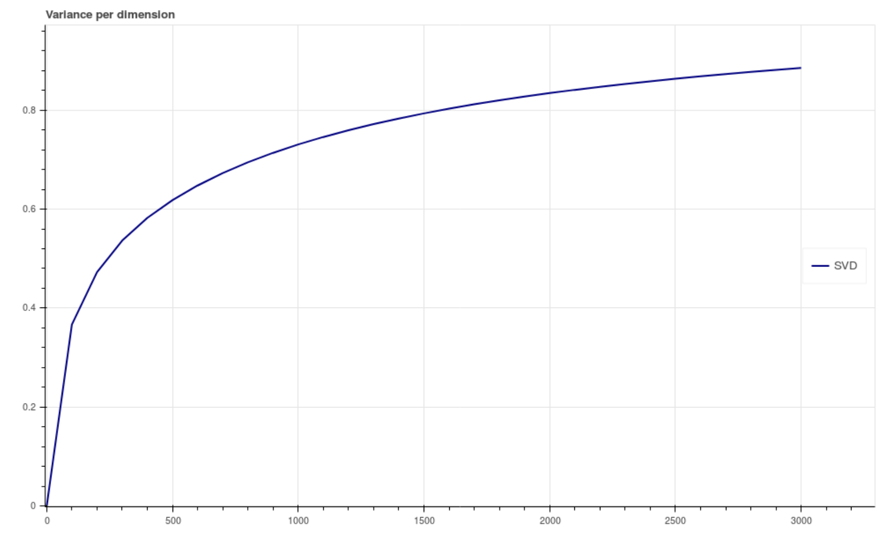


Figure 5.1: Scree plot 1000 dimensions over the entire Nachlass: the x-achsis shows the number of dimensions while the y-achsis represents the maintained variance.

5.2.1 Cluster Algorithm Comparison

Prior to clustering the entire Nachlass, the smaller datasets are examined. Measuring the quality with the external metrics, it can be seen that mainly smaller numbers of clusters are preferred. Table 5.2 shows the ideal number of clusters according to the metrics used. Slashes indicate that two metrics yield divergent results.

Dataset	**K-Means**	**Mean Shift**	**DBSCAN**	**GMM**	**Ward**
Toy Corpus	3	5	3	3	3
Prototractatus	2 / 4	–	2	2	2
Big Typescript	2 / 6	–	2	2	2 / 5
PI	2 / 5	–	2	2	5
Brown Book	2 / 5	–	2	2	2
Kringelbuch	2	3	–	4 / 5	2 / 5
Full Nachlass	300	–	2	250	300

Table 5.2: Compared algorithms and their outcome.

Since no information about the number of clusters is given in advance, the first two algorithms tested are DBSCAN and Mean-Shift. For these two algorithms, no number of clusters k has to be provided by the user. Because the data regions in the entire dataset are very dense, Mean-Shift fails to retrieve any clusters and classifies the data to be one single cluster. This makes it impossible to calculate the evaluation metrics properly. Moreover, clustering the data into a single group will not solve the present problem of making the similarity search more time-efficient. Mean-Shift is therefore excluded from further experiments. Similarly, DBSCAN yields poor results on our dataset, because it only retrieves one cluster and several outliers in the margin regions. Even by merging the outliers to a second cluster, the results are not applicable for boosting performance of WiTTSim.

As for the remaining algorithms tested, the specifying of the number of clusters relies on the user. The number of clusters is however crucial for the clustering outcome quality as well as for the later performance of the similarity search. Moreover, the number is highly dependent on the dataset, which is difficult to determine in an unsupervised setting. The clusters should be of a size such that the performance of the similarity comparison is significantly increased so that it can be provdided in a user-friendly way. The most efficient number of clusters is determined in Figure 5.2, which shows that for 234 clusters, only 468 elements will have to be compared as opposed to comparing 54,930 elements in the original brute-force search.

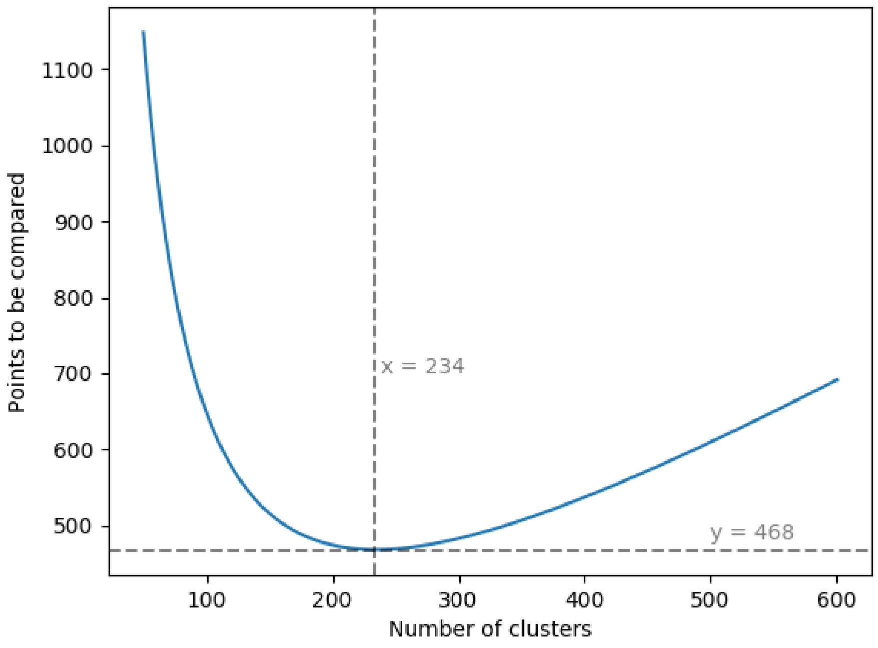


Figure 5.2: The graph shows the number of points that will have to be compared when clustering into the respective number of clusters, assuming a balanced cluster size. In terms of performance, the minimum value of points to be compared yields the ideal cluster size. This means, the best number of clusters is 234 where 468 elements have to be compared.

After obtaining a rough idea of the cluster quantity, the algorithms depending on these numbers can be evlauated. For the K-Means, GMM, and Ward algorithm, the dataset is clustered starting with 50 clusters to 300 by steps of 50. Table 5.3 presents all scores for each of the steps and for all algorithms. The silhouette coefficient determined is equally poor for all tests, yielding results between -0.08 and -0.04. The Calinski-Harabasz score yields the best results with K-Means and $k = 50$, while the Davies-Bouldin index favours the Ward algorithm with 300 clusters. Most interestingly, although the unsupervised evaluation algorithms rate the clustering results fairly poor, all algorithms of all sizes manage to classify the 471 clusters from the labelled corpus correctly. This yields a recall value of 1.00 for each of the algorithms, showing that for our purposes – which is grouping similar remarks together to reduce the search space – any algorithm could be chosen to be integrated. This latter fact is especially crucial for clustering the data prior to the similarity search: Although the clustering step is added beforehand, the overall results do not change. That means, the same top n documents are retrieved regardless whether searching only the clusters or the entire search space. In terms of yielding optimal results, the best clustering algorithm with optimal parameters is found precisely when the exact same results are calculated in both cases:

1. Brute force search: entire data space is searched
2. Cluster-based search: only closest cluster is searched

After evaluating the results on a random basis, the partition into 300 clusters yields too fine-grained results and not all relevant documents could be retrieved. Hence, a smaller value for k should be chosen.

k	**Algorithm**	**Silhouette**	**Calinski-Harabasz**	**Davies-Bouldin**	**Recall**
2	DBSCAN	**0.16**	**1929.07**	–	1.00
50	K-Means	**-0.06**	**210.70**	5.70	1.00
	Ward	-0.09	128.41	**5.62**	1.00
	GMM	**-0.06**	184.48	5.80	1.00
100	K-Means	**-0.05**	**122.07**	4.61	1.00
	Ward	-0.08	79.67	**4.45**	1.00
	GMM	-0.06	106.71	4.74	1.00
150	K-Means	**-0.05**	**90.83**	4.15	1.00
	Ward	-0.08	62.25	**3.78**	1.00
	GMM	-0.06	78.50	4.64	1.00
200	K-Means	**-0.05**	**73.42**	3.88	1.00
	Ward	-0.07	52.91	**3.45**	1.00
	GMM	-0.06	64.66	4.11	1.00
250	K-Means	**-0.04**	**63.43**	3.55	1.00
	Ward	-0.07	47.04	**3.22**	1.00
	GMM	-0.05	56.42	3.43	1.00
300	K-Means	**-0.04**	**56.42**	3.31	1.00
	Ward	-0.07	42.93	**3.09**	1.00
	GMM	-0.05	49.91	3.55	1.00

Table 5.3: Detailed evaluation score results for the entire Nachlass. Dashes indicate that the algorithm was not able to compute the respective score.

For the integration it is indispensable that the algorithm supports a saving of the cluster centres such that the query point can be compared to all centres and, subsequently, the correct cluster can be determined. This excludes the hierarchical Ward algorithm from the integration, leading to two possible algorithms to be examined: K-Means and the GMM. K-Means allows for storing the cluster centroids while equivalently, GMM permits an extraction of the mean values of each cluster. Since the results for K-Means are slightly better than the GMM scores, K-Means in conjunction with SVD will be utilised. As the partition into 300 clusters yields too fine-grained results and not all relevant documents could be retrieved, K-Means with 150 clusters will be integrated.

5.2.2 K-Nearest Neighbour Search

Document clustering is especially useful when determining similar remarks to an unknown input query. When finding similar results to an already existing remark, however, KNN can be integrated, where a clustering beforehand is redundant. This saves memory and time because the query does not have to be located in a certain cluster and then has to be compared to all documents in the cluster. Rather, a data tree is constructed and the most relevant, i.e. the closest, data points are retrieved.

In the KNN experiments the dataset is also reduced to 1,600 features beforehand. Then, for all reduced remarks, the closest k documents are determined using three different parameters: the brute-force KNN, k-d tree, and balltree. All three approaches retrieve the most similar remarks in a reasonable time, while, surprisingly, the brute-force search is the fastest algorithm in our experiments. This may trace back to the fact that the trees have to be built before searching. Pre-saving the k-d tree and the balltree and loading them from disk can make the search even more efficient for the two tree structures. This yields very accurate and fast results, as shown in Table 5.4.

Dataset	**Exhaustive search**	**KD-Tree**	**Balltree**
	(Time in s)	(Time in s)	(Time in s)
Full Nachlass	0.37	6.95	8.23

Table 5.4: Compared KNN performance on different datasets without clustering included.

Also, with further analysis, KNN could be integrated when chosing a non-centroid based algorithm for data exploration and boosting performance by constructing a k-d tree of the entire dataset. Moreover, a classification to the wrong cluster can be avoided. This is a critical point of the above described methods, because if a cluster centroid of data points further away is selected, the results will be misleading and fail to retrieve the closest data points.

5.3 Performance Evaluation

Once the ideal combination of dimensionality reduction and clustering algorithm has been determined, the performance of the new methods extending WiTTSim will be evaluated in three steps. Note that the performance evaluation is only carried out on the entire Nachlass, since comparing a small amount of data points is not only already time-efficient in advance but also not relevant for WiTTSim.

For comparing the performance, the original time was measured. That is, how long does it take to retrieve the top k similar remarks by searching the entire data space and without clustering the remarks beforehand. Secondly, the dimensionalities are drastically reduced from 115,601 to 1,600 features using SVD while maintaining as much information as necessary. Thirdly, a combination of reducing the dimensionalities to $d = 1,600$ using SVD and incorporating the clustering for searching only respective clusters, is presented. The clustering algorithm used is a K-Means algorithm with $k = 150$. Results show that already reducing the features can drastically improve the results from 1940.13 seconds to 17.19. By integrating the clusters, the search time can be further reduced to 5.40 seconds. The runtime comparison of all three experiments is given in Table 5.5.

Dataset	Exhaustive search	Reduced Features	Reduced + clusters
Full Nachlass	1940.13 s (32.34 mins)	17.19 s	5.40 s

Table 5.5: Compared performance on different datasets, before and after clustering.

Depending on the input query, times can vary. For searching the exhaustive space for all features, the required working memory is very high such that the computation had to be outsourced to a high performance computer with 1 terabyte RAM. This emphasises again the necessity of reducing the dimensions such that the calculations can be performed on any system.

Compared to the results retrieved by the KNN algorithm presented in Section 5.2.2 on page 48, the resulting times are still slower using the clustering approach. However, the clustering enables unknown data points to be located in the d-dimensional space and retrieves the most relevant remarks in a reasonable time. Because in many cases, the input query is not part of the database, the clustering algorithm is chosen to be the optimal method of searching and visualising the data. However, for retrieving similar remarks to a remark saved in the database, the KNN search can be helpful. Furthermore, because the most similar results from known remarks will not change, they can be saved in advance and read from file. Due to the limited time frame, however, this is left for future work. All prestored similar remarks are stored in a .csv file and Excel spreadsheet and are provided on the CD attached to this work.

5.4 Challenges

During the different steps of preprocessing, clustering, and integration, several challenges occurred on various levels. Some of these challenges are on technical level, such as reducing the dimensionality and finding the appropriate clustering technique, others are on data level, such as fitting the algorithms onto the Wittgenstein data and considering Wittgenstein data specific properties and needs. The following subsections will describe each of these challenges in further detail.

5.4.1 Curse of Dimensionality

The incorporation of a large number of semantic and syntactic features brings many advantages and opens up new possibilities for retrieving similar remarks. However the high number of dimensions rises also a series of problems and challenges. A major drawback is the complexity: the Nachlass consists of approximately 55,000 remarks with a feature vector of approximately 100,000 features each. This leads to a total number of features of 5.5×10^9 that have to be compared. However not all of these 100,000 features are crucial for determining the similarity among documents. Since the feature space in high dimensions is very sparse, features that occur only once or twice will not contribute to the clustering because they have no significant impact on the entire data space. Therefore, the data space can be reduced drastically while maintaining a high number of distinctive features.

The challenge is to find an appropriate dimensionality reduction method, then reduces to the smallest but best possible number of dimensions. However after enriching the feature vectors with a large number of features, the algorithm should only discard the unnecessary features that do not convey any meaning important for determining cluster similarities. However most algorithms work in an unsupervised way such that it is nearly impossible to survey the reduction process. The problem can be tackled by re-examining the results on a random bases and discarding all reduction methods that fail to keep the variance sufficiently high.

Further, most algorithms rely on a dimensionality reduction method for a closed data set. For projecting the user query onto the same number of dimensions, the parameters of fitting the data have to be saved such that the exact same process can be applied to the incoming user query. Projecting the user query into the wrong data space will lead to the query point being in the wrong data space and the system fails to determine the correct distance in the data space.

5.4.2 Parameter Setting

A vast majority of the algorithm require a manual parameter setting beforehand. In an unsupervised setting, the number of clusters is not known in advance. The number is however crucial for attaining good and satisfying clustering results. This is especially essential in algorithms such as K-Means that require a user-specified seeding. The clustering quality depends on these seeds, since a poor seeding can only converge to a local maximum. Moreover, algorithms such as DBSCAN require a pre-calculation of a parameter ϵ. Similarly, determining the number of clusters k has a high impact on the overall clustering result, where a too large value defines the entire dataset to be one single cluster, and an extremely low value clusters every data point to its own cluster. Summarising, a sensitive choice of user-defined parameters is absolutely indispensable but challenging.

5.4.3 Cluster Structure

Next to choosing appropriate parameters, the resulting cluster structure of each algorithm will also be crucial for considering an integration into WiTTSim. For instance, an algorithm cannot mark data points as outliers. This is because when an outlier is not considered in the clustering list, a user query will never find that data point even if it is exact copy and paste. In other words, the algorithm will be adapted such that it only searches first the cluster centroids and then the cluster points. All data points that do not belong to any cluster will be disregarded which leads to a massive information loss and decrease of accuracy. This excludes certain (well-performing) algorithms to be applicible to our approach, such as DBSCAN or CLIQUE. However the underlying data structure may reveal outliers that will have to be “forced” to belong to a certain cluster even if it is located further away in the data space.

Furthermore, the structure of the Wittgenstein Nachlass in some cases demands for a soft clustering technique. That is, remarks should be allowed to be within two different clusters. This is the case when for example the first paragraph in a remark is similar to remarks a, b and c while the second paragraph resembles remarks d and e. Through all six remarks into the same cluster could be problematic since remark a and remark e may not have a single overlapping neither in syntactic features nor in semantics. To address this issue, remarks will have to be treated on sentence level instead of remark level. An illustration is given in Figure 5.3.

A third challenge in the cluster structure is the inter-cluster distance and the intra-cluster distance. In an artificial corpus, intra-cluster distance is very small while the

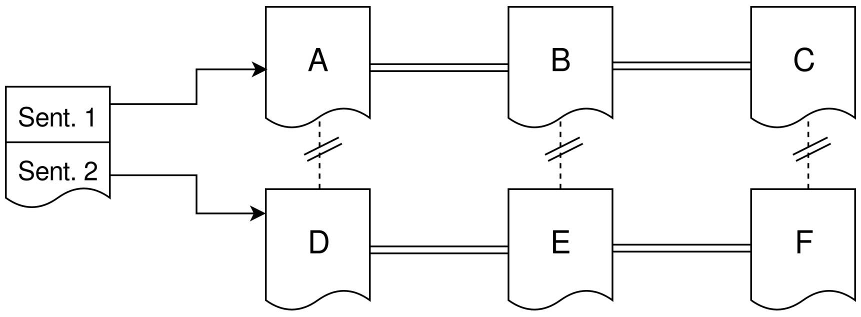


Figure 5.3: Two sentences in the same remark where the first sentences is similar to remarks A, B, and C, while the second sentences shares similar features with remarks D, E, and F. However there is no connection between [A,B,C] and [D,E,F].

inter-cluster distance is large, simply because the clusters are clearly separated from each other. In a more realistic setting, as in the entire Nachlass, the distance between clusters may be relatively small because there is no clear separation or grouping of the points but a more gradual transition between the data points. However the similarity search relies on a well-performing clustering algorithm such that still the closest (and most similar) remarks are retrieved and not discarded simply because they belong to another cluster.

5.5 Discussion

A lot of factors contribute to a functioning clustering algorithm that is sufficiently performant in order to be integrated into the similarity search. A vast amount of dimensionality reduction methods in combination with clustering techniques has been tested and evaluated. The following section will discuss advantages and drawbacks, along with a justification of the integrated methods.

Dimensionality reduction on a large number of features has been proven to be complex for some of the methods presented. PCA was not designed for storing a larger amount of features than the data sample size, which is why it has not been further considered in the experiments. UMAP has proven to produce well-separated clusters, however, the underlying model is too extensive to be stored in a file and be re-used for reducing the input vector. Nevertheless, this is indispensive for mapping the input vector onto the same space as the underlying database. In further studies, UMAP is well-suited for data exploration, while less suitable for our purposes. Next, SRP and SVD have been applied to the original database in order to reduce the database to 532 and 1,600 features, respectively. Although both methods score relatively low in the unsupervised evaluation metrics, SVD seems more promising because of its higher scores in combination with the document clustering.

After reducing the databasis to a reasonable amount of features, the dataset was clustered using five different clustering algorithms. The Mean-Shift algorithm was unable to retrieve any clusters on the SVD-reduced data and on the SRP-reduced data. On the UMAP-reduced data it automatically detects 40 clusters, which shows the importance of choosing a good combination of reduction and clustering algorithm. DBSCAN detects between two and four clusters which is why taking it into account it is negligible in the further experiments. A reduction to only four clusters would still lead to approximately 13,000 data points in a cluster which is too high to be worth considering. K-Means, Ward, and GMM all three yield accurate results, all classifying the small amount of labelled data correctly, leading to a recall value of 1.0. Since only centroid-based algorithms are worth considering for the integration, only K-Means and GMM are possible solutions, where K-Means is chosen for its slightly better evaluation results.

For examining the appropriate number of clusters, clustering sizes between 50 and 300 have been selected for efficiency reasons. A cluster size of 300 seems promising, however, results are already too fine-grained to retrieve satisfying results. Therefore, the final integration is a K-Means clustering algorithm, where $k = 150$ and the data has been reduced to 1,600 features beforehand using SVD.

6 Conclusions and Future Work

The main goal of the current study was to boost performance of the similarity search WiTTSim such that the top k similar remarks are retrieved in a reasonable time. This was achieved in two different ways: Firstly, by incorporating a clustering technique, i.e., clustering the data in advance, comparing the query data point only to the cluster centroids, and subsequently comparing it to all data points in the cluster of the closest centroid. Secondly, for retrieving similar documents to an already known document in the database, the datasets were organised beforehand in form of a k-d tree or balltree, and then only respective branches were searched.

More detailed summaries and outlooks will be given in the following sections: Section 6.1 will recap the obtained results and will bring together the main areas covered in this thesis. Finally, Section 6.2 will address open issues and suggestions for future work.

6.1 Conclusions

Summarising can be captured that selecting an appropriate combination of a dimension reduction algorithm and a clustering technique is indispensible for achieving good results in the similarity search. This is because the performance boost strongly depends on a well-suited combination and, most importantly, not every clustering algorithm is appropriate for every data setting. Combining unsuitable algorithms will fail to detect any clusters at all in the data, and is hence useless to be integrated. For example, density-based algorithms, which determine the number of clusters automatically, were not suitable for the underlying database with 1,600 features after the SVD reduction, and 100 features after the UMAP reduction, respectively. Because of the dense data space, both examined algorithms Mean-Shift and DBSCAN failed to detect the clusters adequatly.

Optimal results could be achieved using K-Means after reducing the dimensions by means of UMAP. However, the dimensionality reduction algorithms need to be suited for saving pre-trained models such that a user query can be fit with the same model and subsequently retrieving the correct projection onto a lower dimensional space. Since the UMAP model is too large to be stored for transforming the query vector, SVD was chosen in the subsequent step, being a simple model that can be stored and restored effectively while at the same time achieving equally good results.

Selecting the features beforehand and diminishing them to a reasonable amount of dimensions had an immense impact on the efficiency of the similarity search. Reducing the features from 115,601 to 1,600 increased the efficiency drastically: The search time could be reduced from 1940.13 seconds to 17.19 seconds. Further, by incorporating clustering additionally, the performance could be even more boosted, handling a user query in solely 5.40 seconds.

For searching similar remarks to already existing remarks in the database, document clustering is not necessary. Instead, the features are again reduced and a KNN search has been implemented to increase performance, retrieving high-quality results in 0.37 up to 8.23 seconds only. The time thereby depends on the organisation of the data, that means, whether the data is searched in a brute-force manner, or if the data is organised as a k-d tree or balltree. Surprisingly, for our dataset, the KNN brute-force approach achieved the fasted results.

6.2 Future Work

Due to the restricted time of this work, only very few algorithms could be tested and still many experiments and implementations remain to be carried out. There is abundant room for further progress in determining the optimal combination between dimensionality reduction method and clustering algorithm by testing even more different algorithms, as described in the following paragraphs.

As an additional reduction technique to be tested, feature agglomeration can be evaluated. Basically, feature agglomeration clusters features by means of hierarchical clustering, and in this way groups together similar features. Future work should examine, how much information can be maintained, while drastically reducing the complexity of the dataset.

Further attempts could be made to implement DNN-driven approaches, such as clustering the data space using a SOM, or implementing a DEC or DCN that have been presented in Chapter 2. Comparing these algorithms with approaches in this work may reveal the optimal baseline for the integration of the clustering as well as for exploratory purposes of the data space.

Moreover, it is desirable for future work to apply the algorithms directly on lower dimensional vectors, for instance retrieving word embeddings from *Word2Vec*, *Doc2Vec*, or *FastText*[1] and entirely skipping the dimension reduction. Furthermore, future research might apply a TF-IDF weighting to the newly obtained vectors in order to weigh the different features accordingly.

[1]see for example `https://radimrehurek.com/gensim/models/word2vec.html`

List of Abbreviations

AMI	Adjusted Mutual Information. 33
ARI	Adjusted Rand Index. 33
BMU	Best Matching Unit. 27
CNN	Convolutional Neural Network. 6, 7
DBSCAN	Density-Based Spatial Clustering of Applications with Noise. 9, 23, 24, 34, 38, 45, 50, 51, 53
DCN	Deep Clustering Network. 7, 8, 54
DEC	Deep Embedded Clustering. 6–8, 54
DNN	Deep Neural Network. 6–8, 54
EM	Expectation Maximization. 22, 23
GMM	Gaussian Mixture Model. 21, 22, 34, 38, 46, 47, 52
KNN	K-Nearest Neighbour. 3, 13, 28, 29, 34, 40, 43, 44, 48, 49, 53, 54, 83
LDA	Linear Discriminant Analysis. 14, 16, 37
LSH	Locality-Sensitive Hashing. 10
LSI	Latent Semantic Indexing. 6
MDL	Minimum Description Length. 21
MI	Mutual Information. 16, 33
MLE	Maximum Likelihood Estimate. 22
NLP	Natural Language Processing. 7, 9
NMI	Normalized Mutual Information. 33
PAM	Partitioning Around Medoids. 20
PCA	Principal Component Analysis. 15, 16, 33, 37, 43, 51, 79
PLSA	Probabilistic Latent Semantic Indexing. 6

POS	Part-of-Speech. 13, 14, 36
RP	Random Projection. 17, 18
SGD	Stochastic Gradiant Descent. 6, 18
SOM	Self-Organizing Map. 8, 9, 26, 27, 54, 79
SRP	Sparse Random Projection. 17, 33, 37, 43, 51, 77
SVD	Singular Value Decomposition. 3, 15, 33, 37, 43, 44, 47, 48, 51–53, 69, 77, 79
t-SNE	t-distributed Stochastic Neighbour Embedding. 17
TF-IDF	Term Frequency-Inverse Document Frequency. 7, 10, 54
UMAP	Uniform Manifold Approximation and Projection. 17, 18, 37, 43, 44, 53, 69, 77
VSM	Vector Space Model. 13, 14
WAST	Wittgenstein Advanced Search Tools. 1, 38
WiTTSim	Wittgenstein Similarity search. 1, 3, 13, 28, 35, 36, 38, 45, 48, 50, 53

Appendices

A Dataset Details

All utilised datasets are parts of Ludwig Wittgenstein's literary remains. While the toy corpus is created artificially for evaluation purposes, all other sets are natural texts without any manipulation required. The remarks can be accessed at the Wittgenstein Ontology Explorer of the Wittgenstein Archives at the University of Bergen (WAB) at `http://wab.uib.no/sfb/` or via the transcriptions at `http://wab.uib.no/transform/wab.php?modus=opsjoner`. The latter also provides information about the language of the texts and the times the texts were published. The table in Table A.1 shows the detailed list or remarks and class labels, if available. For the smaller and mixed datasets, i.e. the toy corpus and the selection from Wittgenstein's Kringelbuch, the remark texts are also provided.

Data Set	**Siglum**	**Class**	**Texts**
Toy Corpus	Ms-165,18[3] et19[1]et20[1]	1	Wir sind durch eine bestimmte Abrichtung, Erziehung, so konditioniert eingestellt , daß wir unter bestimmten Umständen Wunschäußerungen von uns geben. (Ein solcher 'Umstand' ist natürlich nicht der Wunsch.) Eine Frage ob ich weiß was ich wünsche ehe mein Wunsch erfüllt ist kann in diesem Spiele gar nicht auftreten. Und daß ein Ereignis meinen Wunsch zum Schweigen bringt bedeutet in diesem Sinne nicht daß es den Wunsch erfüllt hat. Ich kann z.B. sagen: Ich bin jetzt befriedigt, aber wäre mein Wunsch befriedigt worden, so wäre ich nicht befriedigt. Anderseits wird auch das Wort "Wünschen" so gebraucht: Man sagt "Ich weiß selbst nicht, was ich mir wünsche". H. & D: "Denn die Wünsche verhüllen uns selbst das Gewünschte".)

Data Set	Siglum	Class	Texts
	Ms-129,105[4] et106[1]	1	Wir sind durch eine bestimmte Abrichtung, Erziehung, so eingestellt, daß wir unter bestimmten Umständen Wunschäußerungen von uns geben. (Ein solcher 'Umstand' ist natürlich nicht der Wunsch.) Eine Frage, ob ich weiß, was ich wünsche, ehe mein Wunsch erfüllt ist, kann in diesem Spiele gar nicht auftreten. Und daß ein Ereignis meinen Wunsch zum Schweigen bringt, bedeutet zeigt nicht, daß es die Wunscherfüllung ist. den Wunsch erfüllt. Ich wäre vielleicht nicht befriedigt, wäre mein Wunsch befriedigt worden. Anderseits wird auch das Wort "wünschen" so gebraucht: Man sagt "Ich weiß selbst nicht, was ich mir wünsche". Und in Herman & Dorothea : "Denn die Wünsche verhüllen uns selbst das Gewünschte."
	Ts-230a,64[4] et65[1]	1	Wir sind durch eine bestimmte Abrichtung, Erziehung, so eingestellt, daß wir unter bestimmten Umständen Wunschäußerungen von uns geben. (Ein solcher 'Umstand' ist natürlich nicht der Wunsch.) Eine Frage, ob ich weiß, was ich wünsche, ehe mein Wunsch erfüllt ist, kann in diesem Spiele gar nicht auftreten. Und daß ein Ereignis meinen Wunsch zum Schweigen bringt, bedeutet nicht, daß es den Wunsch erfüllt. Ich wäre vielleicht nicht befriedigt, wäre mein Wunsch befriedigt worden. Anderseits wird auch das Wort "wünschen" so gebraucht: "Ich weiß selbst nicht, was ich mir wünsche." ("Denn die Wünsche verhüllen uns selbst das Gewünschte.") ()

Data Set	Siglum	Class	Texts
	Ts-230b,64[4] et65[1]	1	Wir sind durch eine bestimmte Abrichtung, Erziehung, so eingestellt, daß wir unter bestimmten Umständen Wunschäußerungen von uns geben. (Ein solcher 'Umstand' ist natürlich nicht der Wunsch.) Eine Frage, ob ich weiß, was ich wünsche, ehe mein Wunsch erfüllt ist, kann in diesem Spiele gar nicht auftreten. Und daß ein Ereignis meinen Wunsch zum Schweigen bringt, bedeutet nicht, daß es den Wunsch erfüllt. Ich wäre vielleicht nicht befriedigt, wäre mein Wunsch befriedigt worden. Anderseits wird auch das Wort "wünschen" so gebraucht: "Ich weiß selbst nicht, was ich mir wünsche." ("Denn die Wünsche verhüllen uns selbst das Gewünschte.") ()
	Ts-230c,64[4] et65[1]	1	Wir sind durch eine bestimmte Abrichtung, Erziehung, so eingestellt, daß wir unter bestimmten Umständen Wunschäußerungen von uns geben. (Ein solcher 'Umstand' ist natürlich nicht der Wunsch.) Eine Frage, ob ich weiß, was ich wünsche, ehe mein Wunsch erfüllt ist, kann in diesem Spiele gar nicht auftreten. Und daß ein Ereignis meinen Wunsch zum Schweigen bringt, bedeutet nicht, daß es den Wunsch erfüllt. Ich wäre vielleicht nicht befriedigt, wäre mein Wunsch befriedigt worden. Anderseits wird auch das Wort "wünschen" so gebraucht: "Ich weiß selbst nicht, was ich mir wünsche." ("Denn die Wünsche verhüllen uns selbst das Gewünschte.") ()

Data Set	Siglum	Class	Texts
	Ts-227a,243[3]	1	441. Wir sind von Natur und durch eine bestimmte Abrichtung, Erziehung, so eingestellt, daß wir unter bestimmten Umständen Wunschäußerungen machen. (Ein solcher 'Umstand' ist natürlich nicht der Wunsch.) Eine Frage, ob ich weiß, was ich wünsche, ehe mein Wunsch erfüllt ist, kann in diesem Spiele gar nicht auftreten. Und daß ein Ereignis meinen Wunsch zum Schweigen bringt, bedeutet nicht, daß es den Wunsch erfüllt. Ich wäre vielleicht nicht befriedigt, wäre mein Wunsch befriedigt worden. Anderseits wird auch das Wort "wünschen" so gebraucht: "Ich weiß selbst nicht, was ich mir wünsche." ("Denn die Wünsche verhüllen uns selbst das Gewünschte.") Wie, wenn man fragte: "Weiß ich, wonach ich lange, ehe ich es erhalte?" Wenn ich sprechen gelernt habe, so weiß ich's.
	Ts-227b,243[3]	1	441. Wir sind von Natur und durch eine bestimmte Abrichtung, Erziehung, so eingestellt, daß wir unter bestimmten Umständen Wunschäußerungen machen. (Ein solcher 'Umstand' ist natürlich nicht der Wunsch.) Eine Frage, ob ich weiß, was ich wünsche, ehe mein Wunsch erfüllt ist, kann in diesem Spiele gar nicht auftreten. Und daß ein Ereignis meinen Wunsch zum Schweigen bringt, bedeutet nicht, daß es den Wunsch erfüllt. Ich wäre vielleicht nicht befriedigt, wäre mein Wunsch befriedigt worden. Anderseits wird auch das Wort "wünschen" so gebraucht: "Ich weiß selbst nicht, was ich mir wünsche." ("Denn die Wünsche verhüllen uns selbst das Gewünschte.") Wie, wenn man fragte: "Weiß ich, wonach ich lange, ehe ich es erhalte?" Wenn ich sprechen gelernt habe, so weiß ich's.
	Ms-124,169[6] et170[1]	2	"Mit Zungen reden". Könnte man sich auch denken, daß das die ganze Sprache der Menschen wäre? Wäre so eine Sprache dann ähnlich wie die von Tieren?

Data Set	Siglum	Class	Texts
	Ms-129,173[4] et174[1]	2	Man könnte sich Menschen denken Es könnte Menschen geben , die etwas einer Sprache nicht ganz unähnliches, besäßen: Lautgebärden; ohne Wortschatz oder Grammatik. (‘Mit Zungen reden’?)
	Ts-228,96[3]	2	Man könnte sich Menschen denken, die etwas einer Sprache nicht ganz unähnliches besäßen: Lautgebärden; ohne Wortschatz oder Grammatik. (‘Mit Zungen reden’)
	Ts-230a,11[3]	2	Man könnte sich Menschen denken, die etwas einer Sprache nicht ganz unähnliches besäßen: Lautgebärden, ohne Wortschatz oder Grammatik. (‘Mit Zungen reden’) ()
	Ts-230b,11[3]	2	Man könnte sich Menschen denken, die etwas einer Sprache nicht ganz unähnliches besäßen: Lautgebärden, ohne Wortschatz oder Grammatik. (‘Mit Zungen reden’) ()
	Ts-230c,11[3]	2	Man könnte sich Menschen denken, die etwas einer Sprache nicht ganz unähnliches besäßen: Lautgebärden, ohne Wortschatz oder Grammatik. (‘Mit Zungen reden’) ()
	Ts-227a,268[3] et269[1]	2	528. Man könnte sich Menschen denken, die etwas einer Sprache nicht ganz unähnliches besäßen: Lautgebärden, ohne Wortschatz oder Grammatik. (‘Mit Zungen reden’)
	Ts-227b,268[3] et269[1]	2	528. Man könnte sich Menschen denken, die etwas einer Sprache nicht ganz unähnliches besäßen: Lautgebärden, ohne Wortschatz oder Grammatik. (‘Mit Zungen reden’)
	Ms-109,224[3]	3	Warum die grammatischen Probleme so hart & scheinbar unausrottbar sind – weil sie mit den ältesten Denkgewohnheiten d.h. mit den ältesten Bildern, die in unsere Sprache selbst geprägt sind, zusammenhängen.
	Ts-211,398[2]	3	Warum die grammatischen Probleme so hart und scheinbar unausrottbar sind – weil sie mit den ältesten Denkgewohnheiten, d.h. mit den ältesten Bildern, die in unsere Sprache selbst geprägt sind, zusammenhängen.
	Ts-212,1175[1]	3	39860?2Warum die grammatischen Probleme so hart und anscheinend unausrottbar sind – weil sie mit den ältesten Denkgewohnheiten, d.h. mit den ältesten Bildern, die in unsere Sprache selbst geprägt sind, zusammenhängen. ((Lichtenberg.))

Data Set	Siglum	Class	Texts
	Ts-213,422r[5] et423r[1]	3	Warum die grammatischen Probleme so hart und anscheinend unausrottbar 423bar sind – weil sie mit den ältesten Denkgewohnheiten, d.h. mit den ältesten Bildern, die in unsere Sprache selbst geprägt sind, zusammenhängen. ((Lichtenberg.))
Prototractatus	Ms-104	-	all remarks of Ms-104
Big Typescript	Ts-213	-	all remarks of Ts-213
Philosophical Investigations	Ts-227a, Ts-227b	-	all remarks of Ts-227a and Ts-227b
Brown Book	Ts-310	-	all remarks of Ts-310
Kringelbuch	Ms-107,266[4]	o	(Es ist oft nicht erlaubt in der Philosophie gleich Sinn zu reden, sondern man muß oft zuerst den Unsinn sagen weil man gerade ihn überwinden soll.)
	Ms-107,267[4]et268[1]	o	(Wie man manchmal eine Musik nur im inneren Ohr reproduzieren kann aber sie nicht pfeifen weil das Pfeifen schon die innere Stimme übertönt, so ist manchmal die Stimme eines philosophischen Gedankens so leise daß sie vom Lärm des gesprochenen Wortes schon übertönt wird & nicht mehr gehört werden kann wenn man gefragt wird & reden sprechen soll.)
	Ms-107,269[3]	o	Ich erwarte daß A zur Tür hereinkommt, aber wie wenn es einen Doppelgänger gibt?
	Ms-107,269[4]	o	Zwei Doppelgänger in einem Zimmer die beide das selbe von sich behaupten & mit einander übereinstimmen denn wenn der eine von sich etwas sagt etwa „Ich habe ...", sagt der andere „ganz richtig ich habe ...".
	Ms-108,126[3]	o	Ist es nicht klar daß es bestimmter sein muß zu sagen 26 durch 5 geben den Rest 1 als zu sagen es sei durch 5 nicht teilbar. D.h. ist es damit nicht klar daß ich in gewissem Sinne unbestimmte Sätze in der Arithmetik haben kann?
	Ms-108,126[4]	o	Daß 26 durch 5 nicht teilbar ist kann man ja daraus erkennen daß 26 an der Einerstelle keine 5 hat und hier haben wir wieder einen unbestimmten Satz.

Data Set	Siglum	Class	Texts
	Ms-108,126[5] et127[1]	o	Aber haben wir hier nicht was ich früher einmal sagte daß nämlich die Negation oder die Ungleichungen in der Arithmetik nur in einer gewissen Allgemeinheit auftreten können, denn zu sagen daß 26 an der Einerstelle keine 5 hat scheint doch blödsinnig unmöglich , nicht aber, zu sagen, es stehe hier eine Zahl die nicht 5 als Einerstelle hat.
	Ms-108,130[2]	o	Die Division liefert ein Zahlenpaar. Ist ein Grund einer der beiden Zahlen den Vorzug zu geben? Das heißt in sofern nichts als man nicht statt 13/5 schreiben könnte (2,3) sondern nur 2 + 35.
	Ms-108,130[4]	o	Eine Ungleichung kann so gut auf ihre Richtigkeit geprüft werden, wie eine Gleichung.
	Ms-108,133[4]	o	Wovon drei Striche ein Bild sind, als dessen Bild können sie dienen.
	Ms-107,199[2]	*	Warum nenne ich Zahnschmerzen „meine Zahnschmerzen"?
	Ms-107,199[3]	*	Wenn ich von dem Anderen sage, er habe Zahnschmerzen so meine ich mit „Zahnschmerzen" gleichsam einen Abstrakt von dem was ich gewöhnlich „meine Zahnschmerzen" nenne.
	Ms-107,203[4]	*	Hier trifft man auf das Problem des Wiedererkennens. Wenn ich sage „ich habe jetzt keine Zahnschmerzen werde aber bald welche haben" so setzt das voraus daß ich das Gefühl der Zahnschmerzen als solches wiedererkenne wenn es eintritt.
	Ms-107,203[5]	*	Man könnte das Problem auch so fassen: Mit dem Wort Schmerz meine ich etwas was jetzt nicht existiert. Ist dann das Wort Schmerz nicht Unsinn, es sei denn daß es im Russellschen Sinne eine Beschreibung ist mit Hilfe von Termen die jetzt existieren?
	Ms-107,266[2]	*	Erdichtete Erzählung, gelesenes & gespieltes Theaterstück. Eine erdichtete Erzählung die nicht in der Zeit & im Ort lokiert ist, ist offenbar auf derselben Stufe wie eine falsche die nach Zeit & Ort bestimmt ist. (Märchen, Sage)

Data Set	**Siglum**	**Class**	**Texts**
	Ms-107,271[5]	*	Wenn ich jemanden der Zahnschmerzen hat bemitleide so setze ich mich in Gedanken an seine Stelle. Aber ich setze mich an seine Stelle.
	Ms-107,285[4]	*	Wenn wir plötzlich vom Nebenzimmer in einer uns unbekannten Stimme den Satz „ich habe Zahnschmerzen" hören, so verstehen wir ihn nicht.
Full Nachlass	Ms-101 – Ts-310 (all)	-	

Table A.1: List of test sets and remarks used for the experiments.

B Dimensionality Reduction Results

The number of essential features, i.e. dimensions, highly depends on the type of dataset used. A smaller sample size with a smaller vocabulary will evidently require less dimensions than, say, a large corpus that is rich in vocabulary. The scree plots for each test set show the essential number of dimensions desired for the respective testset. Ideally, the scree plot shows a step in the visualisation to represent the ideal number of features, however, if no step can be detected, at least 80% variance should be maintained. The result for the two smaller test sets, namely the toy corpus and the Kringelbuch, is presented in Figure B.1.

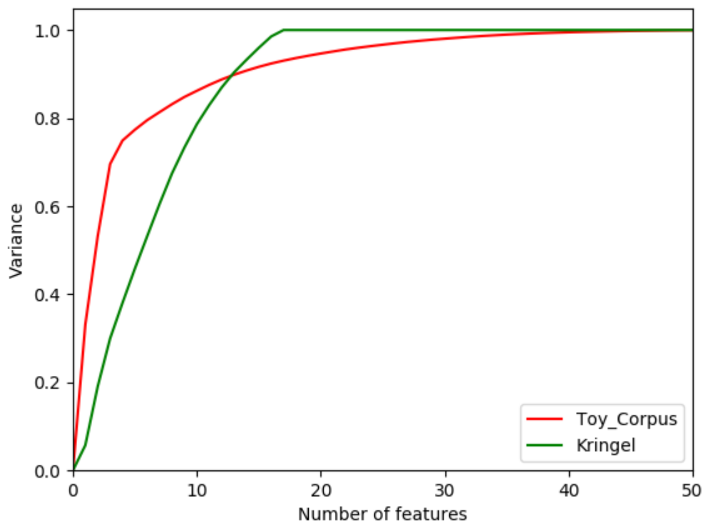


Figure B.1: The scree plot for the artifically created toy corpus and for Wittgenstein's Kringelbuch. The ideal number of features are 36 and 17, respectively. By reducing to the respective number of dimensions, the full variance can be maintained in both cases.

As for the larger datasets, the scree plots are represented in the following. The scree plot for the Prototractatus, the Philosophical Investigations, and the Brown Book is given in Figure B.2 on page 68. For the remaining datasets, namely the Big Typescript and the entire literary remains, i.e. the full Nachlass, Figure B.3 gives an overview of the required number of dimensions.

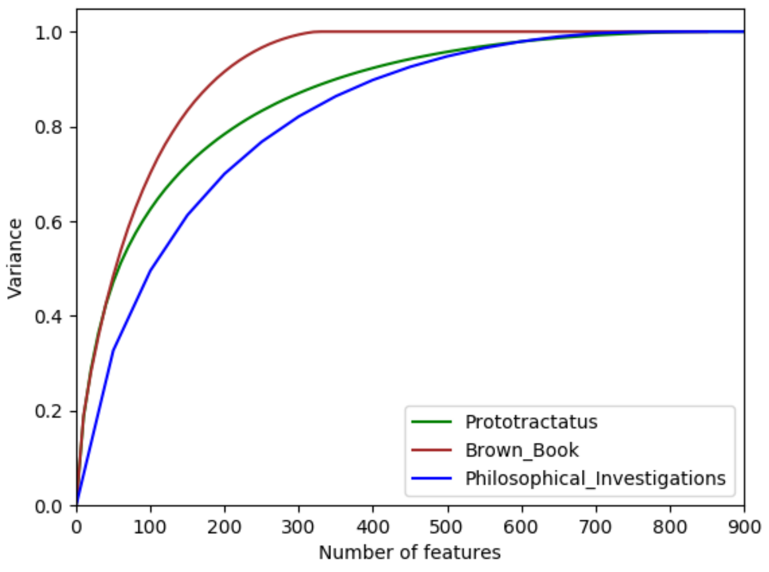


Figure B.2: The scree plot for the Prototractatus, the Brown Book, and the Philosophical Investigations. The smallest dataset, i.e. the Brown Book, should be reduced to 300 dimensions, while the Prototractatus and the Philosophical Investigations can be reduced to 680 and 700 dimensions respectively. In each reduction, a variance of 100% is maintained.

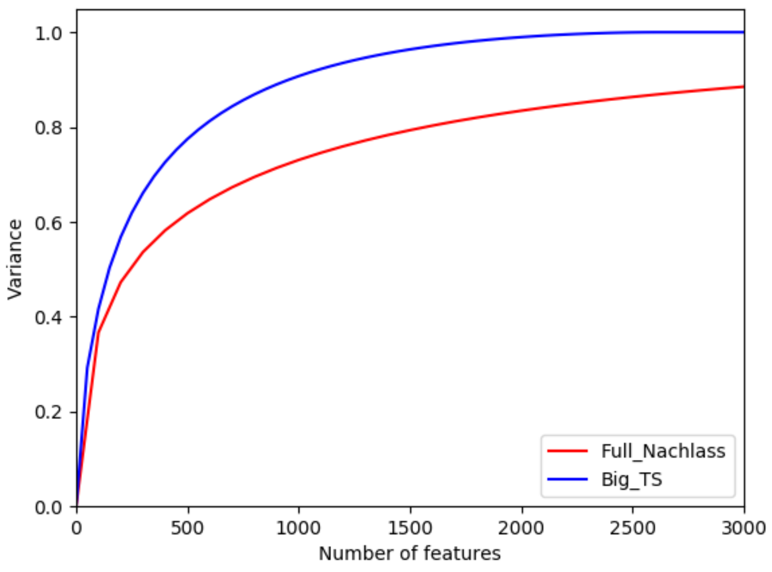


Figure B.3: The scree plot for the two largest datasets, that is, the Big Typescript and the entire Nachlass. Scree plot 1000 dimensions over the full Nachlass. The Big Typescript can be reduced to 2,050 features in order to preserve the entire variance of the data. Due to its complexity, the entire Nachlass is reduced to only 1,600 features maintaining 80% of the original variance. Maintaining the full variance is not efficient, since the curve rises only by 0.1 after each reduction step with a number of dimensions larger than 1,000.

C Clustering Results

This chapter contains the most relevant clustering plots produced in this work. Please note that the data was not clustered in two dimensions. All of the plots however are two-dimensional. This means that although two points look as if they belonged to the same cluster, this is not necessarily the case. They could lie farther away in other dimensions while being close in the first two principal components. For the sake of clarity, this chapter overviews the entire Nachlass, while HTML files are provided on the enclosed CD for all of the clusterings.

Initially, SVD and K-Means clustering with $k = 150$ for the entire Nachlass is illustrated in Figure C.1. The figure shows that using SVD as a dimensionality reduction method, the clusters do not appear well-separated in two-dimensional space. For hovering over the data points and showing the siglums, please refer to the equivalent HTML version on the CD.

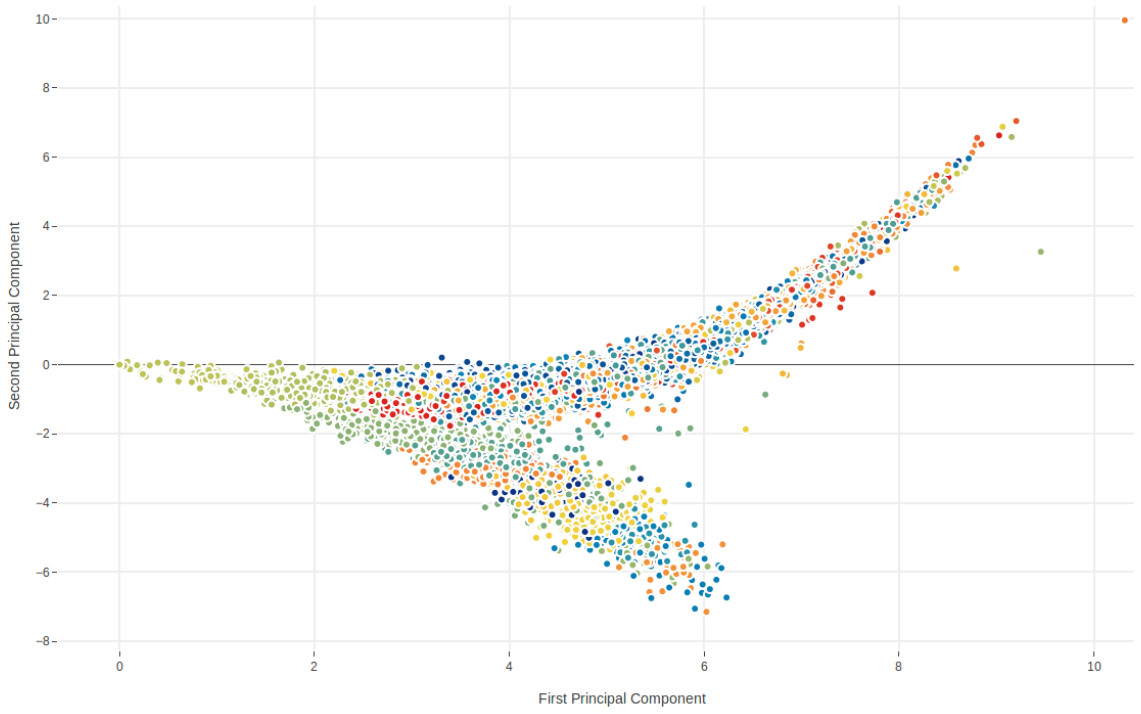


Figure C.1: SVD and K-Means applied on entire dataset for $k = 150$. Although for this combination, the data points are located close in the data space, the cluster structure can be illustrated clearly.

Reducing dimensionalities using UMAP shows the partition into two groups in the visualisation (Figure C.2). For exploratory purposes, Figure C.3 zooms in and shows the the left data group in more detail. Three random clusters are selected, illustrated by the orange, blue, and green box, respectively. Again, the HTML version allows a hovering for more detailed information.

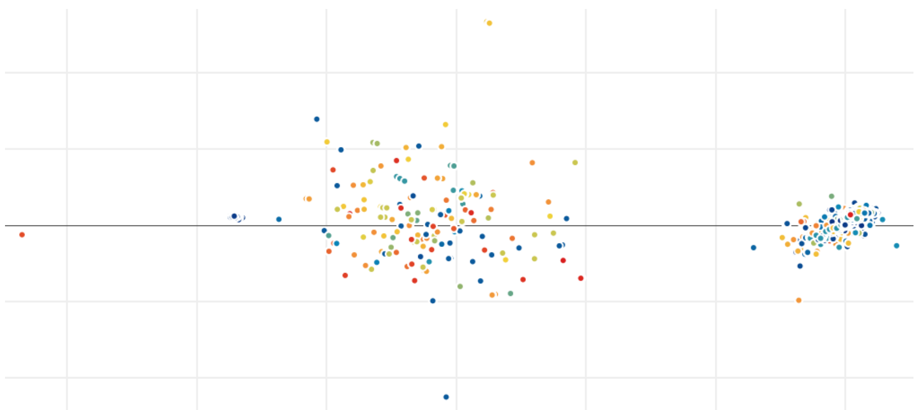


Figure C.2: UMAP and K-Means applied on entire dataset for $k = 150$. The two-dimensional visualisation reveals a data partition into two groups, while further analyses can only be carried out by zooming in.

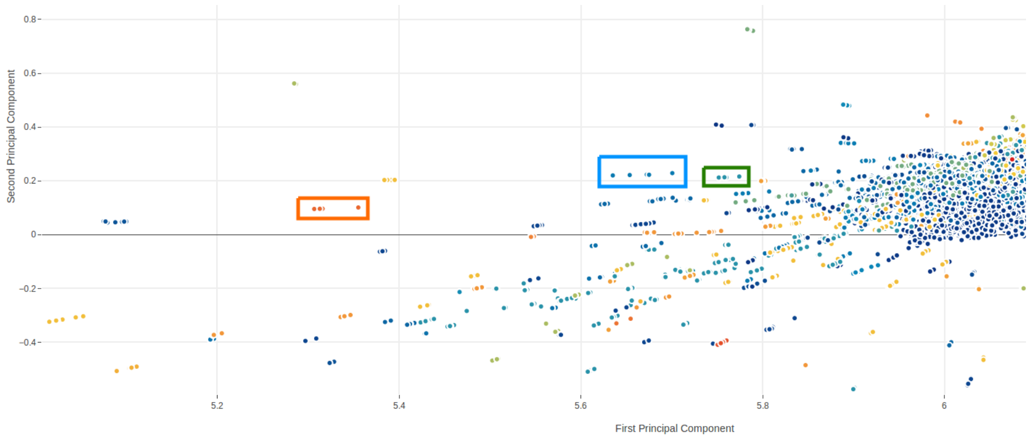


Figure C.3: Selecting three random data groups, represented by differently coloured boxes. The boxes can then be evaluated by examining the texts behind the data points in further detail.

The boxes drawn in Figure C.3 are further described in the following tables. Each table describes one of the boxes and lists the siglums and texts, respectively. Table C.1 shows the results for the green box, while Table C.2 presents the data points that are located in the orange box, and Table C.3 lists remarks in the blue box.

Remarks	**Text**
Ts-228,136[5]	Meine Wahl ist frei, heißt nichts anderes als: ich wähle. Und daß ich manchmal wähle, steht doch nicht im Zweifel. Was man “frei” nennt, ist die Wahl. Zu sagen “Wir glauben nur, daß wir wählen” ist Unsinn. Der Vorgang, den wir “wählen” nennen, findet statt, ob man das Resultat der Wahl sich nach Naturgesetzen voraussagen läßt , oder nicht.

Remarks	Text
Ts-230b,145[3]	Meine Wahl ist frei, heißt nichts anderes, als: ich wähle. Und daß ich manchmal wähle, steht doch nicht im Zweifel. Was man “frei” nennt, ist die Wahl. Zu sagen “Wir glauben nur, daß wir wählen” ist Unsinn. Der Vorgang, den wir “wählen” nennen, findet statt, ob das Resultat der Wahl sich nach Naturgesetzen voraussagen läßt, oder nicht. (
Ts-230c,145[3]	Meine Wahl ist frei, heißt nichts anderes, als: ich wähle. Und daß ich manchmal wähle, steht doch nicht im Zweifel. Was man “frei” nennt, ist die Wahl. Zu sagen “Wir glauben nur, daß wir wählen” ist Unsinn. Der Vorgang, den wir “wählen” nennen, findet statt, ob das Resultat der Wahl sich nach Naturgesetzen voraussagen läßt, oder nicht. (
Ms-115,110[5]et111[1]	Meine Wahl ist frei, heißt nichts anderes als: ich kann wählen wähle manchmal. Und daß ich manchmal wähle, steht doch nicht in Zweifel. Was man “frei” nennt, ist nur die Wahl an sich. Zu sagen, “wir glauben nur, daß wir wählen”, ist Unsinn. Der Vorgang, den 111 wir “wählen” nennen, findet statt, ob man das Resultat der Wahl nach Naturgesetzen vorraussagen kann, oder nicht.
Ms-157a,11r[2]et11v[1]	Meine Wahl ist frei heißt nichts anderes als: ich kann wählen. Und daß ich manchmal wähle, darüber kann doch kein Zweifel sein. Was man frei nennt, ist nur die Wahl an sich. Zu sagen: “wir glauben nur daß wir wählen” ist Unsinn. Der Vorgang den wir ‘wählen’ nennen findet statt ob man das Resultat der Wahl nach Naturgesetzen voraussagen kann oder nicht.

Table C.1: Detailed information about the data points that are located in the green box in Figure C.3.

Remarks	Text
Ts-230b,23[3]	Denken wir, ich fragte: Zeigt es sich uns klar, wenn wir die Sätze aussprechen “Dieser Stab ist 1m lang” und “Hier steht 1 Soldat”, daß wir mit “1” Verschiedenes meinen, daß “1” verschiedene Bedeutungen hat?– Es zeigt sich uns gar nicht. Sag etwa einen Satz, wie “Auf je 1m steht ein Soldat, auf je 2m also 2 Soldaten”. Gefragt, “Meinst du dasselbe mit den beiden Einsern?”– würde man etwa antworten: “Freilich meine ich dasselbe: eins!” (wobei man etwa einen Finger in die Höhe hebt). ()
Ts-230c,23[3]	Denken wir, ich fragte: Zeigt es sich uns klar, wenn wir die Sätze aussprechen “Dieser Stab ist 1m lang” und “Hier steht 1 Soldat”, daß wir mit “1” Verschiedenes meinen, daß “1” verschiedene Bedeutungen hat?– Es zeigt sich uns gar nicht. Sag etwa einen Satz, wie “Auf je 1m steht ein Soldat, auf je 2m also 2 Soldaten”. Gefragt, “Meinst du dasselbe mit den beiden Einsern?”– würde man etwa antworten: “Freilich meine ich dasselbe: eins!” (wobei man etwa einen Finger in die Höhe hebt). ()

Remarks	Text
Ts-230a,23[3]	Denken wir, ich fragte: Zeigt es sich uns klar, wenn wir die Sätze aussprechen “Dieser Stab ist 1m lang” und “Hier steht 1 Soldat”, daß wir mit “1” Verschiedenes meinen, daß “1” verschiedene Bedeutungen hat?– Es zeigt sich uns gar nicht. Sag etwa einen Satz, wie “Auf je 1m steht ein Soldat, auf je 2m also 2 Soldaten”. Gefragt, “Meinst du dasselbe mit den beiden Einsern?”– würde man etwa antworten: “Freilich meine ich dasselbe: eins!” (wobei man etwa einen Finger in die Höhe hebt). ()
Ts-227a,267[4]	“Nachdem er das gesagt hatte, verließ er sie wie am vorigen Tage.”– Verstehe ich diesen Satz? Verstehe ich ihn ebenso, wie ich es täte, wenn ich ihn im Verlaufe einer Mitteilung hörte? Steht er isoliert da, so würde ich sagen, ich weiß nicht, wovon er handelt. Ich wüßte aber doch, wie man diesen Satz etwa gebrauchen könnte; ich könnte selbst einen Zusammenhang für ihn erfinden. (Eine Menge wohlbekannte Pfade führen von diesen Worten aus in alle Richtungen.)
Ts-222,139[2]	Denken wir Denke , ich fragte: Zeigt es sich uns klar, wenn wir die Sätze aussprechen “dieser Stab ist 1m lang” und “hier steht 1 Soldat”, daß wir mit ‘1’ verschiedenes meinen, daß ‘1’ verschiedene Bedeutungen hat? – Es zeigt sich uns garnicht. Besonders, wenn wir einen Satz sagen wie: “Auf je 1m steht 1 Soldat, auf 2m 2 Soldaten u.s.w.”. Gefragt, “ Meinst Du dasselbe mit den beiden Einsern”, würde man etwa antworten: “freilich meine ich dasselbe: – eins!” (wobei man etwa einen Finger in die Höhe hebt).
Ts-228,123[2]	Denken wir, ich fragte: Zeigt es sich uns klar, wenn wir die Sätze aussprechen “Dieser Stab ist 1m lang” und “Hier steht 1 Soldat”, daß wir mit “1” Verschiedenes meinen, daß “1” verschiedene Bedeutungen hat? – Es zeigt sich uns gar nicht. Sag etwa einen Satz wie “Auf je 1m steht ein Soldat, auf je 2m also 2 Soldaten.” Gefragt, “Meinst du dasselbe mit den beiden Einsern?” würde man etwa antworten: “Freilich meine ich dasselbe: eins!” (wobei man etwa einen Finger in die Höhe hebt).
Ts-221a,259[2]	Denken wir, ich fragte: Zeigt es sich uns klar, wenn wir die Sätze aussprechen “dieser Stab ist 1 m lang” und “hier steht 1 Soldat”, daß wir mit ‘1’ verschiedenes meinen, daß ‘1’ verschiedene Bedeutungen hat? – Es zeigt sich uns garnicht. Besonders, wenn wir einen Satz sagen wie: “auf je 1 m steht 1 Soldat, auf 2 m 2 Soldaten u.s.w.”. Gefragt, “meinst Du dasselbe mit den beiden Einsern”, würde man etwa antworten: “freilich meine ich dasselbe: – eins!” (wobei man etwa einen Finger in die Höhe hebt).

Remarks	**Text**
Ms-147,2r[2]et2v[1]	Denken wir ich fragte: “Zeigt es sich uns klar wenn wir die Sätze aussprechen ‘dieser Stab ist 1m lang’ & ‘hier steht 1 Soldat’ daß wir mit ‘1’ verschiedenes meinen, daß ‘1’ verschiedene Bedeutungen hat?” Es zeigt sich uns gar nicht. Besonders wenn wir etwa sagen: “dieses Stück ist 1m lang & es steht ein Soldat hier . . . ”. Gefragt “meinst Du dasselbe” würde man etwa antworten “freilich meine ich dasselbe: eins” (wobei man etwa einen Finger in die Höhe hebt). Meinst Du dasselbe mit “nicht” wenn Du sagst ‘2 2 ist nicht 4’ & ‘dieses Zimmer ist nicht groß’, ‘freilich meine ich dasselbe: nicht!’ (mit einer verneinenden Geste).
Ms-115,63[2]et64[1]	Denken wir, ich fragte: Zeigt es sich uns klar, wenn wir die Sätze aussprechen “dieser Stab ist 1 m lang” & “hier steht 1 Soldat”, daß wir mit ‘1’ verschiedenes meinen, daß ‘1’ verschiedene Bedeutungen hat? – Es zeigt sich uns gar nicht. Besonders Gar, wenn wir einen Satz sagen wie etwa sagen einen Satz sagen wie: “auf je 1 m steht 1 Soldat, auf 2 m 2 Soldaten usw.”

Table C.2: Detailed information about the data points that are located in the orange box in Figure C.3.

Remarks	**Text**
Ts-228,18[4]	Wenn man auch den Satz als Bild eines möglichen Sachverhalts auffaßt und sagt, er zeige die Möglichkeit des Sachverhalts, so kann doch der Satz bestenfalls tun, was ein gemaltes, oder ein plastisches Bild, oder ein Film tut; und er kann also jedenfalls nicht hinstellen, was nicht der Fall ist. Also hängt es ganz von unserer Grammatik ab, was (logisch) möglich genannt wird, und was nicht, – nämlich eben was sie zuläßt? Aber das ist doch willkürlich! – Ist es willkürlich? – Nicht mit jeder satzähnlichen Bildung kann ich etwas anfangen, nicht jedes Spiel ist nützlich, und wenn ich versucht bin, etwas ganz Unnützes als Satz zuzulassen, so geschieht es meistens, weil ich mir seine Anwendung nicht genügend überlegt habe. (“Unendlich lange Baumreihe” – wie ist es zu verifizieren, daß eine solche Reihe unendlich lang ist?).
Ts-230c,42[2]	Wenn man auch den Satz als Bild eines möglichen Sachverhalts auffaßt und sagt, er zeige die Möglichkeit des Sachverhalts, so kann doch der Satz bestenfalls tun, was ein gemaltes, oder ein plastisches Bild, oder ein Film tut; und er kann also jedenfalls nicht hinstellen, was nicht der Fall ist. Also hängt es ganz von unserer Grammatik ab, was (logisch) möglich genannt wird, und was nicht, nämlich eben was sie zuläßt? Aber das ist doch willkürlich!– Ist es willkürlich?– Nicht mit jeder satzähnlichen Bildung kann ich etwas anfangen, nicht jedes Spiel ist nützlich; und wenn ich versucht bin, etwas ganz Unnützes als Satz zuzulassen, so geschieht es meistens, weil ich mir seine Anwendung nicht genügend überlegt habe. (“Unendlich lange Baumreihe” – wie ist es zu verifizieren, daß eine solche Reihe unendlich lang ist?) ()

Remarks	Text
Ts-212,336[2]et337[1]	Wenn man auch den Satz als Bild des beschriebenen Sachverhalts auffaßt & sagt der Satz zeige eben wie es ist, wenn er wahr wäre, er zeige also die Möglichkeit des behaupteten Sachverhalts, so kann der Satz doch bestenfalls tun was ein gemaltes oder modelliertes Bildtun kann tut, & er kann also jedenfalls nicht das hinstellen [erzeugen] was nun eben nicht der Fall ist. a Also hängt es ganz von unserer Grammatik ab was möglich genannt wird & was nicht, nämlich eben, was sie zuläßt. Aber das ist doch willkürlich! – Gewiß, aber nicht mit jedem Gebilde kann ich etwas anfangen; d.h.: nicht jedes Spiel ist nützlich & wenn ich versucht verleitet versucht bin etwas ganz Nutzloses als Satz zuzulassen so geschieht es weilich ich mich durch eine Analogie dazu verleiten lasse & nicht sehe daß mir für meinen Satz noch die wesentlichen Regeln der Anwendung fehlen. So ist es z.B. wenn man von einer unendlichen Baumreihe redet & sich fragt, wie es denn zu verifizieren sei, daß eine Baumreihe unendlich ist & was etwa die Beziehung dieser Verifickation zu der des Satzes „die Baumreihe hat 100 Bäume" ist.
Ts-213,99r[2]et98v[1]	Wenn man auch den Satz als Bild des beschriebenen Sachverhalts auffassßt und sagt, der Satz zeige eben wie es ist, wenn er wahr wäre, er zeige also die Möglichkeit des behaupteten Sachverhalts, so kann der Satz doch bestenfalls tun, was ein gemaltes oder modelliertes Bild tut, und er kann also jedenfalls nicht das hinstellen //erzeugen//, was [Frege] nicht der Fall ist. Also hängt es ganz von unserer Grammatik ab, was möglich genannt wird und was nicht, nämlich eben, was sie zulässßt. Aber das ist doch willkürlich! – Gewissß, aber nicht mit jedem Gebilde kann ich etwas anfangen; d.h.: nicht jedes Spiel ist nützlich und wenn ich versucht bin, etwas ganz Unnützesals Satz zuzulassen, einen Satz zu nennen, so geschieht es, weil ich mich durch eine Analogie dazu verleiten lasse und nicht sehe, dassß mir für meinen Satz noch die wesentlichen Regeln der Anwendung fehlen. Gewiß, aber nicht jeder Kalkül der dem, mit gewissen unserer Erfahrungssätzen, analog ist, ist irgendwie von Nutzen.Nicht jedes Gebilde das in so einem Kalkül jenen Erfahrungssätzen entspricht werden wir Satz nennen wollen. Gewiß aber unsere S Erfahrungssätze z.B. die, welche sich durch ein gemaltes Bild ersetzen ließen weil sie eine sichtbare Verteilung von Körpern beschreiben haben eher eine bestimmte Anwendung einen bestimmten Nutzen. Aber nicht jedes Gebilde das in so einem Kalkül jenen Erfahrungssätzen entspricht werden wir Satz nennen wollen. So ist es z.B., wenn man von einer unendlichen Baumreihe redet und sich fragt, wie es denn zu verifizieren sei, dassß eine Baumreihe unendlich ist, und was etwa die Beziehung dieser Verifikation zu der des Satzes "die Baumreihe hat 100 Bäume" ist.

Remarks	Text
Ms-114,90v[2]	Wenn man auch den Satz als Bild des beschriebenen Sachverhalts auffaßt & sagt, der Satz zeige eben, wie es ist, wenn wie sich die Dinge verhalten, wenn . . . . . . er wahr ist, er zeige also die Möglichkeit des behaupteten Sachverhalts; so kann der Satz doch bestenfalls tun, was ein gemaltes, oder modelliertes, Bild tut, & er kann also jedenfalls nicht das hinstellen, was nun einmal nicht der f Fall ist. Also hängt es ganz von unserer Grammatik ab, was möglich genannt wird & was nicht, nämlich eben was sie zuläßt. Aber das ist doch willkürlich! – Gewiß; aber grammatische Gebilde, welche wir Erfahrungssätze nennen, z.B. die, welche eine sichtbare Verteilung von Körpern im Raum beschreiben & sich durch eine zeichnerische Darstellung ersetzen ließen, haben eine bestimmte Anwendung, einen bestimmten Nutzen. Aber nicht jede Konstruktion, die einem solchen Erfahrungssatz ihrer äußern Form nach, ähnlich ist & die in einem Kalkül eine irgendwie ähnliche Rolle spielt, hateinen analogen Nutzen eine analoge Anwendung, & wir werden dann nicht geneigt sein diese Konstruktion einen Satz zu nennen.

Table C.3: Detailed information about the data points that are located in the blue box in Figure C.3.

All clustering results are provided on the enclosed CD, where the hovering mode allows for easier exploration of the dataspace. For the printed version, the hovering mode is depicted in Figure C.4.

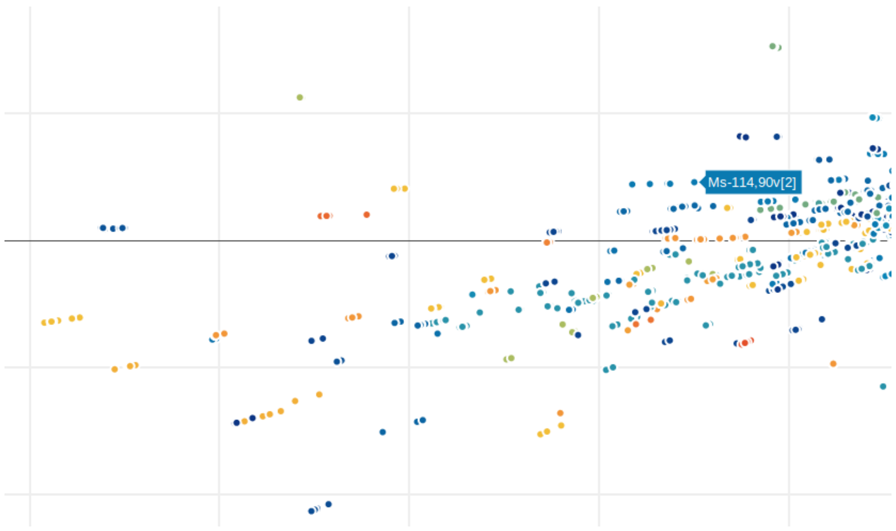


Figure C.4: Hovering over the data points for showing the respective siglums. Here, the selected data point represents Ms-114,90v[2] which is part of the blue box depicted in Figure C.3

D Detailed Scores

In this chapter, detailed results to the reduction techniques combined with the algorithms used will be presented. The evaluation metrics are restricted to the same metrics presented in Chapter 5: SRP, SRP, and UMAP. In order to avoid redundancy, only results not presented in the main part of this work, will be presented. This means, for the evaluation results of SVD in combination with all clustering algorithms, please refer to Table 5.3 on page 47. In this chapter the following results will be given: Table D.1 presents all results obtained by reducing the data with SRP prior to applying the clustering algorithms, while Table D.2 depicts outlines the results with UMAP reduction.

k	**Algorithm**	**Silhouette**	**Calinski-Harabasz**	**Davies-Bouldin**	**Recall**
2	DBSCAN	**0.10**	**1446.93**	5.96	1.00
	Mean-Shift	–	–	–	–
50	K-Means	-0.09	**163.03**	5.36	1.00
	Ward	-0.13	102.17	6.66	1.00
	GMM	-0.09	158.48	5.27	1.00
100	K-Means	-0.09	93.64	4.59	1.00
	Ward	-0.12	64.85	4.94	1.00
	GMM	-0.09	93.45	4.61	1.00
150	K-Means	-0.08	69.73	4.26	1.00
	Ward	-0.12	51.14	4.15	1.00
	GMM	-0.08	69.84	3.81	1.00
200	K-Means	-0.08	56.88	3.74	1.00
	Ward	-0.11	43.70	3.94	1.00
	GMM	-0.08	56.57	3.91	1.00
250	K-Means	-0.08	49.22	3.64	1.00
	Ward	-0.11	38.93	3.62	1.00
	GMM	-0.08	48.51	3.44	1.00
300	K-Means	**-0.07**	43.41	**3.29**	1.00
	Ward	-0.10	35.57	3.33	1.00
	GMM	**-0.07**	42.88	3.36	1.00

Table D.1: Detailed evaluation score results for the entire Nachlass using SRP beforehand as reduction to 523 features. Dashes indicate that the algorithm was not able to separate clusters well and therefore the scores could not be computed. Results printed in bold show the best outcomes for each evaluation measure. For the Calinski-Harabasz index, K-Means is highlighted additionally because the index fails to compute significant values for DBSCAN with $d = 2$.

k	Algorithm	Silhouette	Calinski-Harabasz	Davies-Bouldin	Recall
4	DBSCAN	0.64	**3262.20**	1.66	1.00
60	Mean-Shift	**0.51**	380.23	1.90	1.00
50	K-Means	**0.18**	**1931.46**	2.01	1.00
	Ward	0.12	1816.40	2.54	1.00
	GMM	0.03	1398.04	2.09	1.00
100	K-Means	0.15	1283.31	1.64	1.00
	Ward	0.12	1243.46	1.91	1.00
	GMM	0.03	1008.80	1.78	1.00
150	K-Means	0.17	1131.73	1.35	1.00
	Ward	0.11	1080.38	1.47	1.00
	GMM	0.07	893.29	1.55	1.00
200	K-Means	0.16	1094.02	1.22	1.00
	Ward	0.12	1029.92	1.18	1.00
	GMM	0.06	848.14	1.33	1.00
250	K-Means	0.17	1117.29	1.07	1.00
	Ward	0.13	1024.87	1.08	1.00
	GMM	0.06	857.21	1.22	1.00
300	K-Means	**0.18**	1156.09	**0.99**	1.00
	Ward	0.14	1039.88	1.05	1.00
	GMM	0.08	897.72	1.09	1.00

Table D.2: Detailed evaluation score results for the entire Nachlass with UMAP reduction to 100 features before the clustering. Results printed in bold show the best outcomes for each evaluation measure. For the Calinski-Harabasz index, however, both DBSCAN and K-Means are highlighted because DBSCAN solely separated the data into four clusters, hence the index fails to compute significant values.

List of Figures

List of Tables

CD Content

A ma.pdf

Thesis in PDF format.

B thesis.zip/

All necessary files to generate the pdf file.

C papers.zip/

PDF files of all referenced scientific work.

D code/

Code written for the implementation of the system and all experiments.

1. witt-cluster
    - lib/: contains python modules and the Makefile
    - res/: contains resources used for the experiments
    - out/: contains the generated output
    - README.md: information to run the code
    - Makefile: Makefile for running the code
    - requirements.txt: required python packages. Can be installed by typing `make-install`
2. witt-similar: contains the integration into WiTTSim

Bibliography

[1] D. Achlioptas. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281. ACM, 2001.

[2] D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

[3] C. Afonso, F. Ferreira, J. Exposto, and A. I. Pereira. Comparing clustering and partitioning strategies. In *AIP Conference Proceedings*, volume 1479, pages 782–785. AIP, 2012.

[4] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park. Fast algorithms for projected clustering. In *ACM SIGMoD Record*, volume 28, pages 61–72. ACM, 1999.

[5] C. C. Aggarwal and P. S. Yu. *Finding generalized projected clusters in high dimensional spaces*, volume 29. ACM, 2000.

[6] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[7] S. Al-Anazi, H. Al-Mahmoud, and I. Al-Turaiki. Finding similar documents using different clustering techniques. *Procedia Computer Science*, 82:28–34, 2016.

[8] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*, 2017.

[9] S.-i. Amari and A. Takeuchi. Mathematical theory on formation of category detecting nerve cells. *Biological Cybernetics*, 29(3):127–136, 1978.

[10] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod record*, volume 28, pages 49–60. ACM, 1999.

[11] H. Ashtiani, S. Kushagra, and S. Ben-David. Clustering with same-cluster queries. In *Advances in neural information processing systems*, pages 3216–3224, 2016.

[12] F. Bação, V. Lobo, and M. Painho. Self-organizing maps as substitutes for k-means clustering. In *International Conference on Computational Science*, pages 476–483. Springer, 2005.

[13] E. Bair. Semi-supervised clustering methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(5):349–361, 2013.

[14] S. Balakrishnama and A. Ganapathiraju. Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*, 18:1–8, 1998.

[15] G. H. Ball and D. J. Hall. Isodata, a novel method of data analysis and pattern classification. Technical report, Stanford research inst Menlo Park CA, 1965.

[16] S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer, 2002.

[17] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[18] J. C. Bezdek, C. Coray, R. Gunderson, and J. Watson. Detection and characterization of cluster substructure i. linear structure: Fuzzy c-lines. *SIAM Journal on Applied Mathematics*, 40(2):339–357, 1981.

[19] J. C. Bezdek, R. Ehrlich, and W. Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[20] E. Biçici and D. Yuret. Locally scaled density based clustering. In *International Conference on Adaptive and Natural Computing Algorithms*, pages 739–748. Springer, 2007.

[21] H. Biesenbach and L. Wittgenstein. *Anspielungen und Zitate im Werk Ludwig Wittgensteins*. University of Bergen, 2011.

[22] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001.

[23] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[24] A. Z. Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE, 1997.

[25] Y. Cai and J. Yuan. Text clustering based on improved DBSCAN algorithm. *Computer Engineering*, 12:018, 2011.

[26] T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[27] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(1):5, 2015.

[28] K. S. Candan and M. L. Sapino. *Data management for multimedia retrieval*. Cambridge University Press, 2010.

[29] S.-H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

[30] J.-W. Chang and D.-S. Jin. A new cell-based clustering method for large, high-dimensional data in data mining applications. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 503–507. ACM, 2002.

[31] J.-P. Cheiney and C. de Maindreville. A parallel strategy for transitive closure usind double hash-based clustering. In *VLDB*, pages 347–358, 1990.

[32] C.-H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 84–93, New York, NY, USA, 1999. ACM.

[33] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, K. Olukotun, and A. Y. Ng. Map-reduce for machine learning on multicore. In *Advances in neural information processing systems*, pages 281–288, 2007.

[34] P. Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

[35] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *STOC*, volume 8, pages 537–546. Citeseer, 2008.

[36] S. Dasgupta and A. Gupta. An elementary proof of the johnson-lindenstrauss lemma. *International Computer Science Institute, Technical Report*, 22(1):1–5, 1999.

[37] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[38] D. DeMers and G. W. Cottrell. Non-linear dimensionality reduction. In *Advances in neural information processing systems*, pages 580–587, 1993.

[39] M. Dolatshah, A. Hadian, and B. Minaei-Bidgoli. Ball*-tree: Efficient spatial indexing for constrained nearest-neighbor search in metric spaces. *arXiv preprint arXiv:1511.00628*, 2015.

[40] W. Dong, C. Moses, and K. Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, pages 577–586. ACM, 2011.

[41] A. Dundar, J. Jin, and E. Culurciello. Convolutional clustering for unsupervised learning. *arXiv preprint arXiv:1511.06241*, 2015.

[42] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[43] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[44] K. Fatehi, A. Bozorgi, M. S. Zahedi, and E. Asgarian. Improving semi-supervised constrained k-means clustering method using user feedback. *Journal of Computing and Security*, 1(4):273–261, 2014.

[45] S. Faußer and F. Schwenker. Semi-supervised clustering of large data sets with kernel methods. *Pattern Recognition Letters*, 37:78–84, 2014.

[46] C. Fellbaum. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer, 2010.

[47] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern recognition*, 41(1):176–190, 2008.

[48] E. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–780, 1965.

[49] J. FRIEDMAN. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[50] J. H. Friedman and J. J. Meulman. Clustering objects on subsets of attributes (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(4):815–849, 2004.

[51] J. Gan and Y. Tao. Dbscan revisited: mis-claim, un-fixability, and approximation. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 519–530. ACM, 2015.

[52] S. Goil, H. Nagesh, and A. Choudhary. Mafia: Efficient and scalable subspace clustering for very large data sets. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 443, page 452. ACM, 1999.

[53] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520, 2005.

[54] M. Hadersbeck, A. Pichler, F. Fink, and Ø. L. Gjesdal. Wittgenstein's Nachlass: WiTTFind and Wittgenstein advanced search tools (WAST). In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, pages 91–96. ACM, 2014.

[55] B. Hamp, H. Feldweg, et al. Germanet - A lexical-semantic net for German. In *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15, 1997.

[56] J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[57] V. Henrich and E. W. Hinrichs. GernEdiT-The GermaNet Editing Tool. In *ACL (System Demonstrations)*, pages 19–24. Citeseer, 2010.

[58] A. Hinneburg, D. A. Keim, et al. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, volume 98, pages 58–65, 1998.

[59] T. Honkela. *Self-organizing maps in Natural Language Processing*. PhD thesis, Helsinki University of Technology Espoo, Finland, 1997.

[60] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. WEBSOM—self-organizing maps of document collections. In *Proceedings of WSOM*, volume 97, pages 4–6, 1997.

[61] V. P. Honkela and A. Saarela. Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11(3):1, 2000.

[62] A. Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, volume 4, pages 9–56, 2008.

[63] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[64] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[65] J. A. Kangas, T. K. Kohonen, and J. T. Laaksonen. Variants of self-organizing maps. *IEEE transactions on neural networks*, 1(1):93–99, 1990.

[66] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[67] T. Kohonen. Essentials of the self-organizing map. *Neural networks*, 37:52–65, 2013.

[68] T. Kohonen, S. Kaski, P. Somervuo, K. Lagus, M. Oja, and V. Paatero. Biennial Report 2002-2003, chapter 8. *CIS, February*, pages 113–122, 2004.

[69] H.-P. Kriegel and M. Pfeifle. Density-based clustering of uncertain data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 672–677. ACM, 2005.

[70] D. Kriesel. *A Brief Introduction to Neural Networks.* 2007.

[71] P. Kröger. Knowledge Discovery in Databases II. University Lecture, LMU München, May 2019.

[72] S. Kushagra, S. Ben-David, and I. Ilyas. Semi-supervised clustering for deduplication. *arXiv preprint arXiv:1810.04361*, 2018.

[73] A. Ławrynowicz. *Semantic Data Mining: An Ontology-Based Approach*, volume 29. IOS Press, 2017.

[74] E. G. Learned-Miller. Entropy and mutual information. *Department of Computer Science, University of Massachusetts, Amherst*, 2013.

[75] L. Lee. Measures of distributional similarity. *arXiv preprint cs/0001012*, 2000.

[76] C. Li and B. Wang. Principal components analysis. 2014.

[77] P. Li, T. J. Hastie, and K. W. Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296. ACM, 2006.

[78] T. Li, M. Ogihara, and G. Tzanetakis. *Music data mining.* CRC Press, 2011.

[79] M. Lindinger. Entwicklung eines WEB-basierten Faksimileviewers mit Highlighting von Suchmaschinen-Treffern und Anzeige der zugehörigen Texte in unterschiedlichen Editionsformaten. Master's thesis, LMU, 2015.

[80] B. Liu, Y. Xia, and P. S. Yu. Clustering through decision tree construction. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 20–29. ACM, 2000.

[81] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[82] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[83] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[84] A. F. McDaid, D. Greene, and N. Hurley. Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint arXiv:1110.2515*, 2011.

[85] L. McInnes, J. Healy, and S. Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11), mar 2017.

[86] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[87] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[88] G. Miller. *WordNet: An electronic lexical database.* MIT press, 1998.

[89] J. Nayak, B. Naik, and H. Behera. Fuzzy c-means (fcm) clustering algorithm: a decade review from 2000 to 2014. In *Computational intelligence in data mining-volume 2*, pages 133–149. Springer, 2015.

[90] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[91] M. Oja, S. Kaski, and T. Kohonen. Bibliography of self-organizing map (som) papers: 1998-2001 addendum. *Neural computing surveys*, 3(1):1–156, 2003.

[92] S. M. Omohundro. *Five balltree construction algorithms.* International Computer Science Institute Berkeley, 1989.

[93] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *Acm Sigkdd Explorations Newsletter*, 6(1):90–105, 2004.

[94] U. Patki, S. Kishor, and P. Khot. Fuzzy Document Clustering based on Frequent Features and Feature Length. 2018.

[95] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

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

[97] D. Pelleg and A. Moore. Accelerating exact k-means algorithms with geometric reasoning. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 2000.

[98] D. Pelleg, A. W. Moore, et al. X-means: extending k-means with efficient estimation of the number of clusters. In *Icml*, volume 1, pages 727–734, 2000.

[99] A. Pichler. *Untersuchungen zu Wittgensteins Nachlaß*. The Wittgenstein Archives, 1994.

[100] A. Pichler, H. Krüger, D. Smith, T. Bruvik, A. Lindebjerg, and V. Olstad, editors. *Wittgenstein Source Bergen Facsimile (BTE)*. Wittgenstein Source Bergen, 2009.

[101] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. Murali. A monte carlo algorithm for fast projective clustering. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 418–427. ACM, 2002.

[102] M. O. Rabin. Fingerprinting by random polynomials. *Technical report*, 1981.

[103] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[104] D. Ravichandran, P. Pantel, and E. Hovy. Randomized algorithms and nlp: Using locality sensitive hash functions for high speed noun clustering. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 622–629, 2005.

[105] R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the em algorithm. *SIAM review*, 26(2):195–239, 1984.

[106] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[107] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of statistics*, pages 416–431, 1983.

[108] H. Ritter and T. Kohonen. Self-organizing semantic maps. *Biological cybernetics*, 61(4):241–254, 1989.

[109] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.

[110] J. Rothhaupt. Wittgensteins Kringel-Buch. Ludwig-Maximilians-Universität München, Fakultät für Philosophie, Wissenschaftstheorie und Religionswissenschaft, 2011.

[111] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[112] P. J. Rousseeuw and L. Kaufman. Finding groups in data. *Hoboken: Wiley Online Library*, 1990.

[113] I. Röhrer. *Musik und Ludwig Wittgenstein: Semantische Suche in seinem Nachlass*. Bachelor's thesis, LMU, 2017.

[114] I. Röhrer, S. Ullrich, and M. Hadersbeck. Weltkulturerbe international digital: Erweiterung der Wittgenstein Advanced Search Tools durch Semantisierung und neuronale maschinelle Übersetzung. *multimedial multimodal. Abstracts zur*

Jahrestagung des Verbandes Digital Humanities im deutschsprachigen Raum, 25. - 29.03.2019 an den Universitäten zu Mainz und Frankfurt, 2019.

[115] C. Sadowski and G. Levin. SimHash: Hash-based similarity detection. *Technical report, Google*, 2007.

[116] J. M. Santos and M. Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *International conference on artificial neural networks*, pages 175–184. Springer, 2009.

[117] A. Schmidt. Ludwig Wittgenstein's Nachlass in the UNESCO Memory of the World register. *Nordic Wittgenstein Review*, 7(2):209–213, 2018.

[118] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[119] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)*, 42(3):19, 2017.

[120] H. Schütze and C. Silverstein. Projections for efficient document clustering. 1997.

[121] T. Seidl. Knowledge Discovery and Data Mining I. Unsupervised Methods - Clustering. University Lecture, LMU München, Mar 2019.

[122] H. Shen and C.-Z. Xu. Hash-based proximity clustering for efficient load balancing in heterogeneous dht networks. *Journal of Parallel and Distributed Computing*, 68(5):686–702, 2008.

[123] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *TextMining Workshop at KDD2000 (May 2000)*, 2000.

[124] H. Steinhaus. Quelques applications des principes topologiques à la géométrie des corps convexes. *Fund. Math*, 41:284–290, 1955.

[125] A. Strehl and J. Ghosh. Cluster ensembles-a knowledge reuse framework for combining partitionings. In *Aaai/iaai*, pages 93–99, 2002.

[126] A. Struyf, M. Hubert, P. Rousseeuw, et al. Clustering in an object-oriented environment. *Journal of Statistical Software*, 1(4):1–30, 1997.

[127] M. A. Tan, A. Meleqi, and A. Berasategui. Informationsverarbeitung II, NLP Group. Seminar work, Ludwig-Maximilians-Universität München, 2018.

[128] J. Tang, J. Liu, M. Zhang, and Q. Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th international conference on world wide web*, pages 287–297. International World Wide Web Conferences Steering Committee, 2016.

[129] M. C. Thrun. *Projection-based clustering through self-organization and swarm intelligence: combining cluster analysis with the visualization of high-dimensional data*. Springer, 2018.

[130] K. Torkkola. Linear discriminant analysis in document classification. In *IEEE ICDM Workshop on Text Mining*, pages 800–806. Citeseer, 2001.

[131] J. D. Ullman. Mining Massive Datasets, Chapter 11. University Lecture, Stanford University, 2017. `http://infolab.stanford.edu/~ullman/mmds/ch11.pdf`.

[132] S. Ullrich, D. Bruder, and M. Hadersbeck. Aufdecken von "versteckten" Einflüssen: Teil-Automatisierte Textgenetische Prozesse mit Methoden der Computerlinguistik und des Machine Learning. *Kritik der digitalen Vernunft. Abstracts zur Jahrestagung des Verbandes Digital Humanities im deutschsprachigen Raum, 26.02.-02.03. 2018 an der Universität zu Köln, veranstaltet vom Cologne Center for eHumanities (CCeH)*, 2018.

[133] L. Van Der Maaten, E. Postma, and J. Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

[134] J. Vesanto, E. Alhoniemi, et al. Clustering of the self-organizing map. *IEEE Transactions on neural networks*, 11(3):586–600, 2000.

[135] R. Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011.

[136] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th annual international conference on machine learning*, pages 1073–1080. ACM, 2009.

[137] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[138] S. Wagner and D. Wagner. *Comparing clusterings: an overview.* Universität Karlsruhe, Fakultät für Informatik Karlsruhe, 2007.

[139] M. J. Warrens. On the equivalence of cohen's kappa and the hubert-arabie adjusted rand index. *Journal of Classification*, 25(2):177–183, 2008.

[140] T. Wei, Y. Lu, H. Chang, Q. Zhou, and X. Bao. A semantic approach for text clustering using WordNet and lexical chains. *Expert Systems with Applications*, 42(4):2264–2275, 2015.

[141] Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 975–982. IEEE, 1999.

[142] D. J. Willshaw and C. Von Der Malsburg. How patterned neural connections can be set up by self-organization. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 194(1117):431–445, 1976.

[143] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2016.

[144] K.-G. Woo, J.-H. Lee, M.-H. Kim, and Y.-J. Lee. Findit: a fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, 46(4):255–271, 2004.

[145] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[146] J. Xu, W. Peng, T. Guanhua, X. Bo, Z. Jun, W. Fangyuan, H. Hongwei, et al. Short text clustering via convolutional neural networks. 2015.

[147] R. Xu and D. C. Wunsch. Survey of clustering algorithms. 2005.

[148] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3861–3870. JMLR. org, 2017.

[149] J. Yang, W. Wang, H. Wang, and P. Yu. δ-clusters: capturing subspace correlation in a large data set. In *Proceedings 18th international conference on data engineering*, pages 517–528. IEEE, 2002.

[150] J. Ye, R. Janardan, and Q. Li. Two-dimensional linear discriminant analysis. In *Advances in neural information processing systems*, pages 1569–1576, 2005.

[151] S. Yu. *Advanced probabilistic models for clustering and projection.* PhD thesis, LMU, 2006.

[152] J. Zamora, M. Mendoza, and H. Allende. Hashing-based clustering in high dimensional data. *Expert Systems with Applications*, 62:202–211, 2016.

[153] J. Zhang. *Visualization for Information Retrieval*, volume 23. Springer Science & Business Media, 2007.

[154] T. Zhang, P. Ji, M. Harandi, R. Hartley, and I. Reid. Scalable deep k-subspace clustering. *arXiv preprint arXiv:1811.01045*, 2018.

[155] W. Zhang, X. Tang, and T. Yoshida. TESC: An approach to text classification using semi-supervised clustering. *Knowledge-Based Systems*, 75:152–160, 2015.

[156] P.-Y. Zhou and K. C. Chan. A model-based multivariate time series clustering algorithm. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 805–817. Springer, 2014.