Unsupervised Estimation of Subjective Content Descriptions in an Information System

Magnus Bender[*], Tanya Braun[†], Ralf Möller[*], and Marcel Gehrke[*]

[*] *Institute of Information Systems, University of Lübeck,*
Ratzeburger Allee 160, Lübeck, Germany
{bender, moeller, gehrke}@ifis.uni-luebeck.de
http://www.ifis.uni-luebeck.de

[†] *Computer Science Department, University of Münster,*
Einsteinstr. 62, Münster, Germany
tanya.braun@uni-muenster.de
http://www.uni-muenster.de/Informatik.AGBraun

Let us consider the following scenario: A human is working with a corpus of text documents. In this corpus, the human needs to know documents with similar content and highlight relevant locations in retrieved documents. An information system displaying the contents of the corpus and providing an information retrieval agent will help the human. To perform information retrieval on the corpus, the agent used internally in the information system may need additional data associated with the documents. In order to support this, so-called Subjective Content Descriptions (SCDs) provide additional location-specific data for text documents. SCDs are subjective in the sense that the agent associates data with sentences to reflect beliefs of users. In our scenario, the agent needs SCDs referencing sentences of similar content across various documents in the corpus and most text documents are not associated with SCDs. Therefore, this paper presents UESM, the Unsupervised Estimator for SCDs Matrices, an approach to associate any corpus with SCDs. In an evaluation, we show that the performance of UESM in estimating topics of similar content in the corpus is on par with latent Dirichlet allocation, while UESM provides SCDs referencing sentences of similar content.

Keywords: Subjective content descriptions, text annotation, topic modelling, sentence clustering, information system.

1. Introduction

An agent in pursuit of a task, explicitly or implicitly defined, may work with a corpus of text documents as a reference library. From an agent-theoretic perspective, an agent is a rational, autonomous unit acting in a world and fulfilling a defined task, e.g., providing document retrieval services given requests from users. Such an agent may be integrated into an information system to be accessed by humans via a graphical user interface (GUI) and by other applications via an application programming interface (API).

We assume that the user, a human or another application, provides a corpus to the information system. This corpus then represents the context of the task defined by the user of the information system, since document retrieval is not an end in itself. Further, documents in a given corpus might be associated with additional location-specific data making the content nearby the location explicit by providing descriptions, references, or explanations. We refer to these additional location-specific data as Subjective Content Descriptions (SCDs) [1]. SCDs are subjective in the sense that the agent associates data with text parts, e.g., sentences, to reflect beliefs of the user.

Associating documents with SCDs support agents in the task of information retrieval (IR). Returning to the information system containing a document retrieval agent, it is valuable for the agent to have for each sentence a set of references to similar sentences in the documents across its corpus. Using these references, the system can show to the users a set of similar sentences for each sentence they are currently reading. Such references can be modeled by SCDs being associated with the sentences. More precisely, an SCD matrix—built on the corpus associated with SCDs—is used to model the SCDs and their locations in the documents of the corpus. The SCD matrix consists of a row for each SCD and each row contains a word distribution of the referenced sentences. In our understanding, each SCD represents a concept or topic mentioned in the corpus. Each SCD's concept is implicitly defined by the word distribution and the content of the sentences referenced by each SCD. Thus, the sentences of an SCD are similar.

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

However, most corpora are not associated with SCDs nor contain references to sentences representing the same concepts. In a first step, we are interested to identifying similar sentences—preferably in an unsupervised manner. Then, using the identified similar sentences, we form SCDs, where each SCD represents a different concept in the corpus and references multiple locations in the text documents of the corpus.

As a solution to the lack of SCDs for most corpora, this paper presents Unsupervised Estimator for SCD Matrices (UESM), an approach to estimate an SCD matrix for any corpus in an unsupervised manner. Thus, UESM associates any corpus of text documents with SCDs. Mainly, UESM detects similar concepts referenced in the text documents of the corpus and then forms an SCD, which groups all occurrences of the same concept.

The remainder of this paper is structured as follows: First, we conclude the introduction with applications of SCDs and describe the analogies between SCD matrices and topic models. Afterwards, we recap the basics of SCDs, specifically of the SCD matrix. We then describe the problem of estimating SCDs in an unsupervised manner and our solution UESM. We introduce three methods for UESM, namely a greedy, a K-Means [2], and a DBSCAN [3] based method. An information system has to automatically select one best method and the best hyperparameters. Thus, we provide a model selection approach for SCD matrices. In the end, we compare UESM with its three methods in an evaluation against the well-know Latent Dirichlet Allocation (LDA) [4] and we present an information system based on UESM. Finally, we look at related work and conclude afterwards.

In addition to the applications of SCDs just described, SCDs can support agents by performing the following tasks:

(i) Estimating SCDs for a single previously unseen text document using the Most Probably Suited SCD (MPS^{2}CD) algorithm [5],
(ii) classifying a text document as related, extended, revised, or unrelated to a corpus [5],
(iii) moving the SCDs from one corpus to another similar corpus by adapting the SCDs' domain [6],
(iv) separating SCDs and actual content being interleaved in text documents [7],
(v) enriching SCDs in a corpus already sparsely associated with SCDs [8], or
(vi) detecting complementary documents to a corpus [9].

Common to all of the above approaches is the need to start with a corpus associated with SCDs. With UESM, we can lift the previous assumption, that we need SCDs to start with. Thus, we do not need some type of supervised learning to get an initial set of SCDs and can get around Wiktionary[a] or OpenIE [10] needed for the evaluations of [7] or [6], respectively.

Summarized, UESM allows for estimating an SCD matrix and thus SCDs for any corpus. Hence, UESM enables the above described approaches to be applied to any corpus without the need for SCDs in the corpus beforehand, as UESM provides the required SCD matrix. In the context of the information system, where the users may provide any corpus with which the system then has to work, UESM can be used to estimate an initial SCD matrix for the corpus. Then, the SCDs can be used by the integrated agent to provide IR services for the users of the system.

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

[a] `https://www.wiktionary.org/`

2. Preliminaries

Before we dedicate to UESM, this section specifies notations and recaps the basics of SCDs.

2.1. *Notations*

First, we formalize our setting of a corpus.

- A word w_i is a basic unit of discrete data from a vocabulary $\mathcal{V} = \{w_1, \ldots, w_L\}$, $L \in \mathbb{N}$.
- A sentence s is defined as a sequence of words $s = (w_1, \ldots, w_N)$, $N \in \mathbb{N}$, where each word $w_i \in s$ is an element of vocabulary $\mathcal{V}$. Commonly, a sentence is terminated by punctuation symbols like ".", "!", or "?".
- A document d is defined as a sequence of sentences $d = (s_1^d, ..., s_M^d)$, $M \in \mathbb{N}$.
- A corpus $\mathcal{D}$ represents a set of documents $\{d_1, \ldots, d_D\}$, $D \in \mathbb{N}$.
- An SCD t is a tuple of the SCD's additional data $\mathcal{C}$ and the referenced sentences $\{s_1, ..., s_S\}$, $S \in \mathbb{N}$. Thus, each SCD references sentences in documents of $\mathcal{D}$, while in the opposite direction a sentence is associated with an SCD.
- A sentence associated with an SCD is called SCD window, inspired by a tumbling window moving over the words of a document. Generally, an SCD window might not be equal to a sentence and may be a subsequence of a sentence or the concatenated subsequences of two sentences, too. Even though, in this paper, an SCD window always equals a sentence.
- For a corpus $\mathcal{D}$ there exists a set g called SCD set containing K associated SCDs

$$g(\mathcal{D}) = \left\{ t_j = \left(\mathcal{C}_j, \bigcup_{d \in \mathcal{D}} \{s_1^d, ..., s_S^d\} \right) \right\}_{j=1}^{K}.$$

  Given a document $d \in \mathcal{D}$, the term $g(d)$ refers to the set of SCDs associated with sentences from document d.
- Each word $w_i \in s^d$ is associated with an influence value $I(w_i, s^d)$ representing the relevance of w_i in the sentence s^d. For example, the closer w_i is positioned to the object of the sentence s^d, the higher its corresponding influence value $I(w_i, s^d)$. The influence value is chosen according to the task and might be distributed binomial, linear, or constant.

2.2. *Subjective Content Descriptions*

SCDs provide additional location-specific data for documents [1]. The data provided by SCDs may be of various types, like additional definitions or links to knowledge graphs. However, in this paper we do not focus on the additional data, instead we focus on how to determine which sentences belong to one SCD.

Algorithm 1 Supervised estimation of SCD matrices $\delta(\mathcal{D})$

function BUILDMATRIX($\mathcal{D}$, $g(\mathcal{D})$)
    Input: Corpus $\mathcal{D}$; Set of SCDs $g(\mathcal{D})$
    Output: SCD-word distribution matrix $\delta(\mathcal{D})$
    Initialize an $K \times L$ matrix $\delta(\mathcal{D})$ with zeros
    for each document $d \in \mathcal{D}$ **do**
        for each SCD $t = (\mathcal{C}, \{s_1^d, ..., s_S^d\}) \in g(d)$ **do**
            for $j = 1, ..., S$ **do** ▷ Iterate over sentences
                for each word $w_i \in s_j^d$ **do**
                    $\delta(\mathcal{D})[t][w_i] \mathrel{+}= I(w_i, s_j^d)$
    return $\delta(\mathcal{D})$

Kuhr et al. use an SCD-word distribution represented by a matrix when working with SCDs [1]. The SCD-word distribution matrix, in short SCD matrix, can be interpreted as a generative model. A generative model for SCDs is characterized by the assumption that the SCDs generate the words of the documents. We assume that each SCD shows a specific distribution of words of the referenced sentences in the documents.

Before we describe UESM, we outline the details of SCD matrices and an algorithm training an SCD matrix $\delta(\mathcal{D})$ for a corpus $\mathcal{D}$ given the SCD set $g(\mathcal{D})$ in a supervised manner.

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

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

The SCD matrix consists of K rows, one for each SCD in $g(\mathcal{D})$, and each row contains the word probability distribution for the SCD. Therefore, the SCD matrix has L columns, one for each word in the vocabulary of the corresponding corpus.

The supervised estimation of an SCD matrix is described in Algorithm 1. Given a corpus $\mathcal{D}$, the algorithm iterates over each document d in the corpus and the document's SCDs. For each associated SCD t, the referenced sentences $s_1^d, ..., s_S^d$ are used to update the SCD matrix. Thereby, the row of the matrix representing SCD t gets incremented for each word in each sentence by each word's influence value.

Finally, the SCD matrix needs to be normalized row-wise to meet the requirements of a probability distribution. However, we skip the normalization because multiple calculations on small decimal values on a computer reduce the accuracy.

Later, we use the cosine similarity with the rows of the matrix and the cosine similarity does a normalization by definition. Thus, by skipping the normalization, we save computational resources and get slightly more accurate results.

Next, we present UESM, which estimates an SCD matrix $\delta(\mathcal{D})$ without needing the SCD set $g(\mathcal{D})$.

3. Unsupervised Estimation of SCDs

The unsupervised estimation of SCDs is divided into two parts. First, an SCD matrix needs to be estimated for a corpus. Given an SCD matrix, the SCDs of a corpus are defined by their SCD-word distributions and the referenced sentences. For UESM, however, there are three methods with multiple hyperparameters resulting in multiple estimated SCD matrices for each corpus. Thus, to use UESM in an information system, we introduce a model selection approach for SCD matrices, afterwards.

3.1. *Unsupervised Estimation of SCD Matrices*

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

Algorithm 2 outlines UESM. The input of UESM is a corpus of which it has to estimate the SCD matrix. Commonly, a sentence is associated with an SCD and each SCD references one or multiple sentences. UESM initially starts by associating each sentence to one unique SCD. The SCD's word distribution of each SCD then only contains the words of the referenced sentence. Lines 10 - 14 of Algorithm 2 show how to create this initial SCD matrix, which consists of a row for each sentence in the document's corpus. The word distributions are calculated using the influence value the same way as in Algorithm 1.

The next step is to find the sentences that represent the same concept and group them into one SCD. There are three different methods for detecting similar rows in the initial SCD matrix. Lines 16 - 33 of Algorithm 2 show the three methods and how the rows are merged. If there are more than two rows, two are merged at a time until all are merged. The main idea of merging two rows is to sum up the quantities of each word in both distributions of words and deleting the second row from the matrix.

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

Algorithm 2 Unsupervised Estimator for SCD Matrices $\delta(\mathcal{D})$

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

Greedy by Similarity The first method greedily selects the next two rows to merge. It calculates the cosine similarity between all rows, containing the word distributions, in the matrix and always merges the two most similar rows. This is repeated until the similarity between the two most similar rows is below the threshold θ (Algorithm 2 lines 17 - 21). Thus, with a lower threshold less SCDs with more referenced sentences each will be estimated and a higher threshold leads to more SCDs with less referenced sentences.

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

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

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

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

K-Means One well-know clustering technique is K-Means [2]. We will not get into the details how K-Means works, but focus on how to apply K-Means. K-Means is initialized with K centroids of which each centroid represents a cluster. Each point is assigned the nearest centroid in terms of the Euclidean distance using a vector representation of the point. Iteratively, the clusters are optimized by aligning each centroid in the center of all the points contained in the centroid's cluster.

We run K-Means on the rows of the SCD matrix to detect clusters of similar rows in the initial SCD matrix. Each row represents a point and the word distribution is the vector representation of this point. After K-Means is finished, we merge the rows of the matrix included in the same cluster (Algorithm 2 lines 27 - 33). Hence, the number of clusters is equal to the number of SCDs in the end. As hyperparameter, the number of SCDs to estimate K is specified. Alternatively, K can be specified by a factor to multiply with the initial number of sentences in the corpus, e.g., the factor 0.25 sets the number of SCDs to a quarter of the sentences in the corpus. Furthermore, there are technique to estimate a *good* number of cluster for K-Means on the corpus [11].

DBSCAN Another well-know clustering technique is DBSCAN [3]. In contrast to K-Means, DBSCAN is able to detect concave structures in data and works density based. DBSCAN clusters two points together if both are in a neighborhood, the distance making up a neighborhood is defined by the hyperparameter ε. A cluster

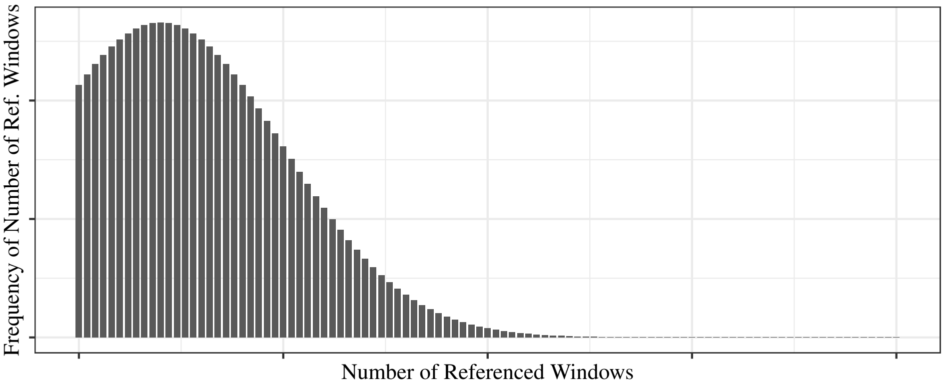

Figure 1. Desired optimal distribution of the number of referenced windows for an SCD matrix. A histogram depicting the different numbers of windows referenced in a SCD matrix should show a similar course.

then grows by adding all points in the neighborhood to the same cluster. Additionally, there is a minimum samples threshold ms which defines the minimum number of points needed to form a cluster.

We run DBSCAN on the cosine similarity matrix $S_{\delta(\mathcal{D})}$ and again merge the rows of the matrix included in the same cluster (Algorithm 2 lines 27 - 33).

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

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

3.2. *Model Selection for SCD Matrices*

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

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

Each SCD references a number of windows and we can use these numbers of references to measure the quality of a matrix. We argue, that a good SCD references

Algorithm 3 SCD Matrix Model Selection

function ESTIMATEBESTMATRIX($\mathcal{D}$)
  Input: Corpus $\mathcal{D}$
  Output: *Best* SCD-word distribution matrix $\delta(\mathcal{D})$
  $sim_{best} \leftarrow 0$
  $\delta_{best} \leftarrow Nil$ ▷ Iterate all methods
  for each method $m \in \{\texttt{Greedy}, \texttt{K-Means}, \texttt{DBSCAN}\}$ **do**
    ▷ Take a set of hyperparameters depending on method
    if $m = \texttt{Greedy}$ **then**
      $H \leftarrow (0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1)$ ▷ Values of θ
    if $m = \texttt{K-Means}$ **then**
      $M' \leftarrow \sum_{d \in \mathcal{D}} M^d$ ▷ Number of sentences in $\mathcal{D}$ to calculate Ks
      $H \leftarrow (\lfloor M' \cdot 0.8 \rfloor, \lfloor M' \cdot 0.6 \rfloor, \lfloor M' \cdot 0.4 \rfloor, \lfloor M' \cdot 0.3 \rfloor, \lfloor M' \cdot 0.2 \rfloor, \lfloor M' \cdot 0.1 \rfloor)$
    else
      $H \leftarrow ((0.3, 1), (0.5, 10), (0.5, 5), (0.5, 2), (0.7, 10))$ ▷ Tuples of ε, ms
    for each hyperparameter $h \in H$ **do**
      $\delta(\mathcal{D}) \leftarrow \text{UESM}(\mathcal{D}, m, h)$ ▷ Run UESM
      ▷ Calculate score using Hellinger distance to normal distribution
      $sim \leftarrow 1 - \text{HD}(\text{SCALE}([0, 100], \delta(\mathcal{D})), \mathcal{N}([0, 100], \mu = 10, \sigma^2 = 15))$
      if $sim > sim_{best}$ **then**
        $sim_{best} \leftarrow sim$
        $\delta_{best} \leftarrow \delta(\mathcal{D})$
  return δ_{best}

a smaller amount of windows, i.e., a reference to 100 or more similar windows in the corpus is less benefcial for a human working with the corpus than a reference to fewer windows. Additionally, an SCD referencing only one or two windows is not really beneficial either. Based on these deliberations, we assume that the distribution of the number of referenced windows for an SCD matrix shown in Figure 1 is optimal. Similar to a histogram, on the x-axis the number of referenced windows for the SCDs are shown and on the y-axis the desired frequencies of each number of windows are shown. We omit the values on the axes because the actual values are not of relevance here—the course of the graph is the crucial point. For the sake of completeness: the graph shows a discretized normal distribution with mean 10 and a standard derivation of 15 in the interval from 0 to 100.

Returning to the quality score of a matrix, we need a possibility to compare the distribution of different numbers of windows referenced in an SCD matrix with the assumed optimal normal distribution. Therefore, we scale the distribution of different numbers of windows to the interval from 0 to 100. Furthermore, we discretize the normal distribution. Afterwards, the distance of both distributions can be calculated with the Hellinger distance [12]

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

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

The entire model selection approach is described by Algorithm 3. The algorithm takes a corpus and returns the best SCD matrix. In Line 6 it starts with iterating over all three methods of UESM. Depending on the method in Lines 8 - 14 different sequences of hyperparameter H to try are chosen. These hyperparameters shall cover a wider range of possibly good hyperparameters for the corpus. Through the for loop starting in Line 15 for each method and the previously chosen hyperparameter H UESM estimates an SCD matrix for corpus $\mathcal{D}$. Afterwards, in Line 18 the resulting SCD matrix is scored using the Hellinger distance after scaling the result as described above. Finally, Line 22 returns the SCD matrix resulting in the highest quality score.

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

4. Evaluation

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

4.1. *Dataset*

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

[b]`https://www.gesetze-im-internet.de/bgb/`, Englisch translation `https://www.gesetze-im-internet.de/englisch_bgb/`

The entire corpus consists of 2 462 law paragraphs and overall 8 020 sentences which are used as SCD windows. Each law paragraph contains between 1 and 45 sentences with an average of 3.3 sentences. The vocabulary consist of 5 294 words, where each sentence is between 1 and 51 with an average of 10.9 words long.

4.2. *Metrics*

Topic models are trained unsupervised using statistical methods, thus, the topics gained by LDA are statistically optimized but may not match human judgement of *good* topics. In general, automatically evaluating the quality of a model from a human point of view is a difficult task. A common measure to evaluate the interpretability of topics regarding human judgement is coherence. Röder et al. [13] compare and evaluate multiple coherence measures against human judgement as gold standard. The authors gain the best results using the C_V measure. However, due to negative correlations and problems reproducing the C_V values in their paper, Röder does not recommended to use the C_V coherence any more[c]. Therefore, in our evaluation we use the UMass coherence calculated using Gensim's coherence model.

As already stated in Subsection 3.2, the number of referenced SCD windows per SCD is relevant. For example, having 1 000 SCD windows and 100 SCDs, each SCD should have a similar number around 10 referenced SCD windows. It would be bad, if 99 SCDs reference 1 window each and the 1 remaining SCD references the remaining 901 windows. Therefore, we evaluate the number of referenced windows per SCD. Besides showing all numbers of referenced windows, we also show the numbers only for SCDs with two or more referenced windows, i.e., we interpret SCDs with only one referenced window as an irrelevant SCD to omit.

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

4.3. *Workflow and Implementation*

UESM is implemented using Python and runs inside a Docker container. The implementation uses the libraries Gensim[d], NumPy[e], and NLTK[f]. The evaluation of UESM follows this workflow:

(i) Extract the law paragraphs from the BGB's XML file and divide each paragraph into its sentences, which are then used as initial SCD windows.
(ii) Lowercase all characters, tokenize the sentences into words, stem the words, and eliminate stop words from a wordlist containing 232 German words. These four tasks are called preprocessing tasks. Preprocessing a text of a document

[c] "The usage of the C_V coherence is not recommended anymore!", stated on `https://github.com/dice-group/Palmetto/wiki/How-Palmetto-can-be-used`, last accessed 24. September 2022
[d] `https://radimrehurek.com/gensim/`
[e] `https://numpy.org/`
[f] `https://www.nltk.org/`

transforms the text in a more digestible form for machine learning algorithms and increases their performance [14].

(iii) Form an initial SCD matrix where each row contains the word probability distribution for one sentence of the corpus.

(iv) Apply UESM with one of the three methods greedy, K-Means, or DBSCAN to detect similar rows in the SCD matrix. Afterwards, merge the similar rows or the rows in the same cluster by summing the distributions' values.

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

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

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

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

LDA Documents This topic model is trained on the 2 462 law paragraphs as documents and applies LDA in its typical fashion with medium sized documents. However, using this model's document topic distributions it is not possible to determine the topic of each sentence, as each of the model's documents contain more than one sentence.

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

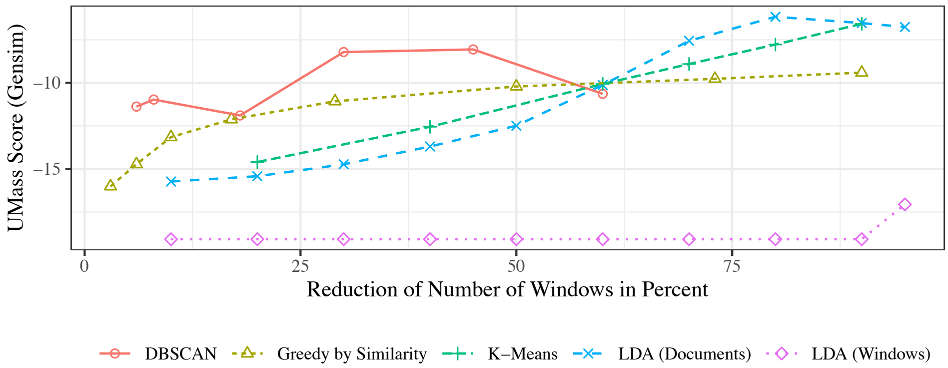

Figure 2. UMass coherence of the three methods using UESM and the coherences of both topic models trained using LDA for comparison.

4.4. *Results*

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

In Figure 2, the coherences of the three methods using UESM and both topic models are shown. The UMass scores calculated by Gensim are negative, higher values are better. On the left side, the reduction of the number of windows is small, thus many SCDs are created. Going to the right, the number of SCDs decreases, e.g., the rightmost triangle of greedy similarity represents 834 SCDs gained from initially 8 020 windows.

The lines of DBSCAN, greedy similarity, K-Means, and LDA Documents are all close together, while LDA Windows shows poor results far below all other lines. This demonstrates that LDA Windows is not capable of estimating SCDs in an unsupervised manner, because the windows used as documents are too small. LDA Documents however demonstrates the UMass score a good topic model reaches on the BGB and UESM using K-Means reaches similarly good values. UESM works well with greedy similarity and less reduction of windows, but K-Means becomes better for more reduction. DBSCAN is quite unstable and the amount of reduction is difficult to configure using the hyperparameters ε and number of minimum samples ms. Although, the coherences of DBSCAN are good, we later see in Figure 3 why DBSCAN is not a good choice.

To summarize, using UESM with K-Means yields coherences on par with LDA. However, LDA is not able to estimate SCDs what UESM does.

In Figure 3 for each of the three methods two plots are shown. In the upper row, for each percentage of reduction the numbers of referenced windows are shown by boxplots on a logarithmic scale. The lower row shows the same, but SCDs referencing

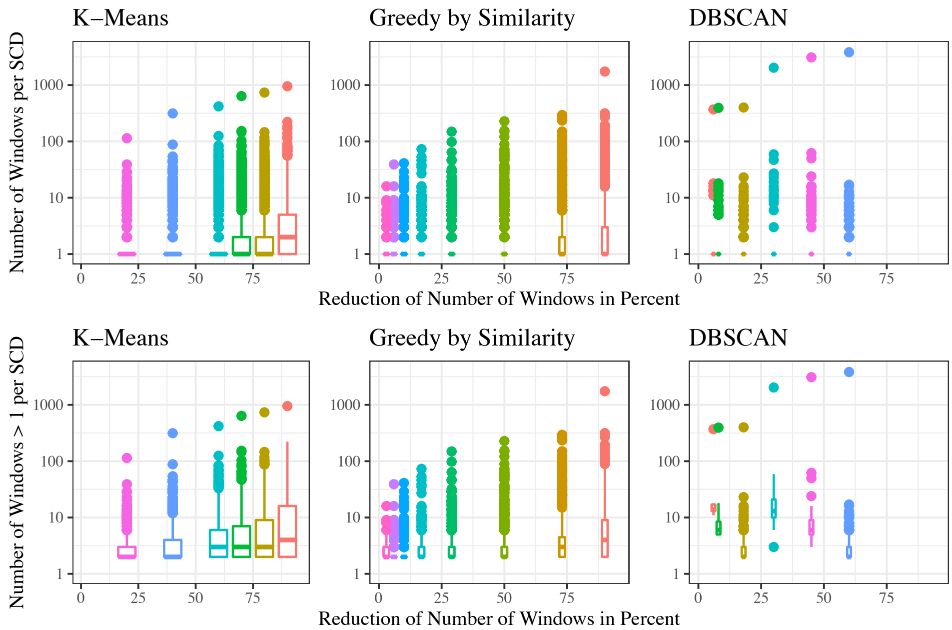

Figure 3. Number of windows referenced by one SCD for the three different methods of UESM. In the lower row, SCDs referencing only one window are omitted.

only one window are omitted. We focus on the lower row: For K-Means and greedy similarity most SCDs reference less than 10 windows, which is a good number of references. However, there are also many outliers referencing more windows. For K-Means the largest number of references is 952 and 1 741 with greedy similarity. An SCD referencing 1 741 windows references 21 % of the corpus and it is hard to imagine that 21 % of the corpus share the same concept. Again, this demonstrates that greedy similarity does not work well with a high reduction of the number of windows.

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

Summarized, K-Means shows an overall very good distribution of referenced windows per SCD and greedy similarity is good, too. Though, DBSCAN generates an SCD referencing nearly half of the corpus. Thus, DBSCAN should not be used.

Next, we present the information system which provides an information retrieval service using UESM for user supplied corpora.

5. Information System

This section describes the information system using UESM. The information system provides an interface for humans and other applications to use UESM to analyze user-submitted corpora. Internally, the information system uses UESM and SCDs to retrieve documents of similar contexts given queries from users. First, we describe the basic structure and demonstrate afterwards how the system can assist a user.

5.1. *Basic Structure*

The information system is a web application which consists of an web interface written in HTML, CSS, and JavaScript. On the server side, the system is written in Python using FastAPI[g] and runs inside a Docker container. The system stores all corpora and models, consisting of the estimated SCD matrices by UESM, on the server.

The web interface is used to manage corpora and navigate through the contained text document and estimated SCDs. Also, the system provides an SCD-based search using the MPS^{2}CD algorithm [5]. In addition to the HTML based GUI, which can be used by humans, there exists a JSON API. Using the API, other applications can send queries to UESM through the information system, without requiring any knowledge of UESM and information retrieval techniques. The API of our information system is used as source of information for a humanoid service robot. Thereby, the humanoid service robot provides a human-friendly way of interaction with information retrieval agents and thus brings technology closer to the people [15].

The system is capable of processing corpora in different languages, including English and German. In addition to law texts, the system can process other corpora of text documents provided as plain-text or PDF documents. We demonstrate the features of the system using the example of the BGB.

5.2. *Working with Corpora*

Let us assume, *Charlie* wants to find similarities and similar paragraphs in the german civil code, the BGB. First, *Charlie* opens the web interface of our information system in a web browser and uses username and password for authentication. In our case, the corpus BGB is already available in the system and thus *Charlie* can directly choose to view the BGB.

If a corpus is not available in the system, the corpus may be imported by uploading a zip archive containing either a XML file, multiple plain-text documents, or PDF documents. The text documents in the zip archive then represent the corpus which UESM uses to estimate the SCD matrix for. After UESM has finished, the corpus can be viewed by *Charlie*.

Each content, e.g., law paragraph or page of a PDF document, is visualized as shown in Figure 4. Additionally to this content itself, buttons to navigation to

[g] `https://fastapi.tiangolo.com/`

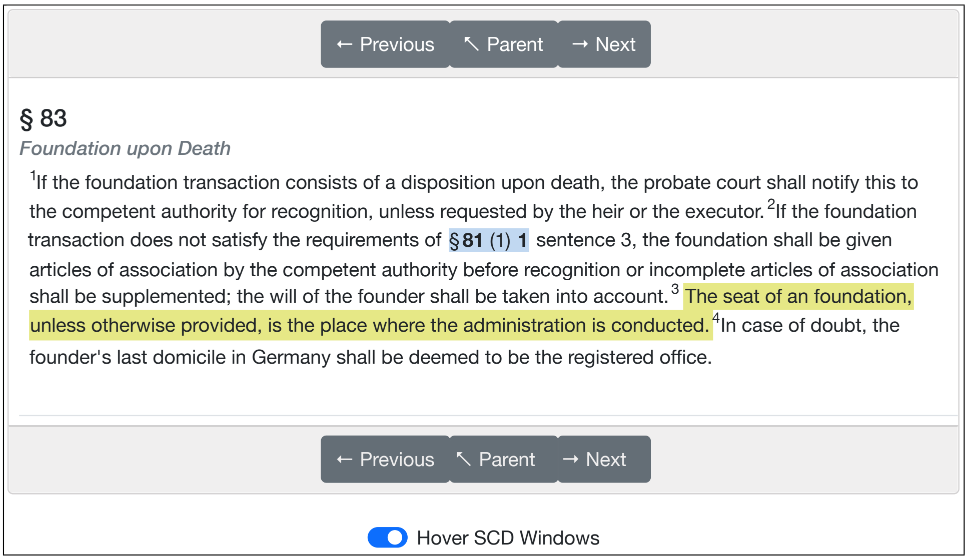

Figure 4. A paragraph or page of the corpus shown in the web interface. The buttons below and above the content allow to navigate forwards and backwards. SCD windows are highlighted yellow on hover.
For the demonstration purposes, we exchange the original German texts with their English translations. The system supports English corpora, but there exists no English XML file for the BGB.

the previous and next paragraph are available. The SCD windows are highlighted yellow on hover, i.e., when the mouse is moved over them. Doing a double-click on a highlighted window opens the assigned SCD.

Most corpora are divide into multiple sections and thereby provide some type of structure to depict. In Figure 5, a larger area of the web interface visualizing corpora is shown. On the left side of the content, a table of contents is available and can be used to jump to different sections. Above the content, a small bar shows the location of the currently shown content as path through the structure of the corpus. In the upper right corner, an input box for queries to search the corpus is available.

Assume, *Charlie* reads different paragraphs of the BGB and is then interested in similar paragraphs to the third sentence of § 83 (highlighted in Figure 4). Hence, *Charlie* does a double-click on the highlighted sentence and *Charlie's* web browser opens the assigned SCD to the just double-clicked SCD window.

5.3. *Using SCDs*

After doing the double-click, *Charlie's* web browser visualizes the selected SCD. The SCD was previously estimated by UESM and the used method and hyperparameters were determined by the model selection approach. However, the user *Charlie* notices

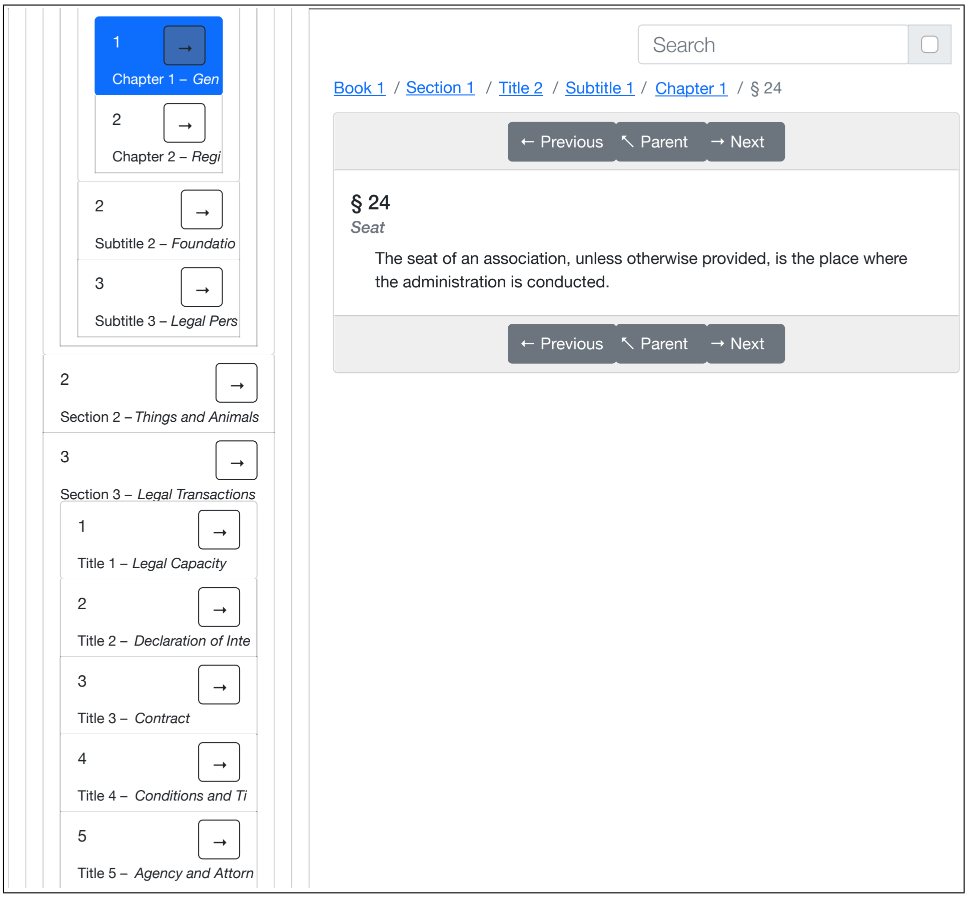

Figure 5. View the documents of a corpus in the information system via the web interface. On the left side a table of contents is shown, while on the middle of the right side content is shown, again. In the upper right corner a full text search for the corpus is available.

nothing about these internal steps.

Figure 6 shows the visualization of an SCD. First, the most probable words of the SCD-word distribution are shown and below the referenced sentences are listed. The list does not only consist of a the referenced sentences of the SCD. Each referenced sentence is shown as excerpt of the corpus together with its surrounding content while the SCD window itself is again highlighted yellow. Showing the surrounding sentences is more human friendly and allows *Charlie* to grasp the context of the referenced and similar sentences more quickly. Thus, *Charlie* can identify the most relevant sentences for *Charlie's* information need and choose to open a corpus' paragraph or SCD window using the blue or gray buttons left of the excerpt.

If *Charlie* is not satisfied with identified similar sentences, it also possible to

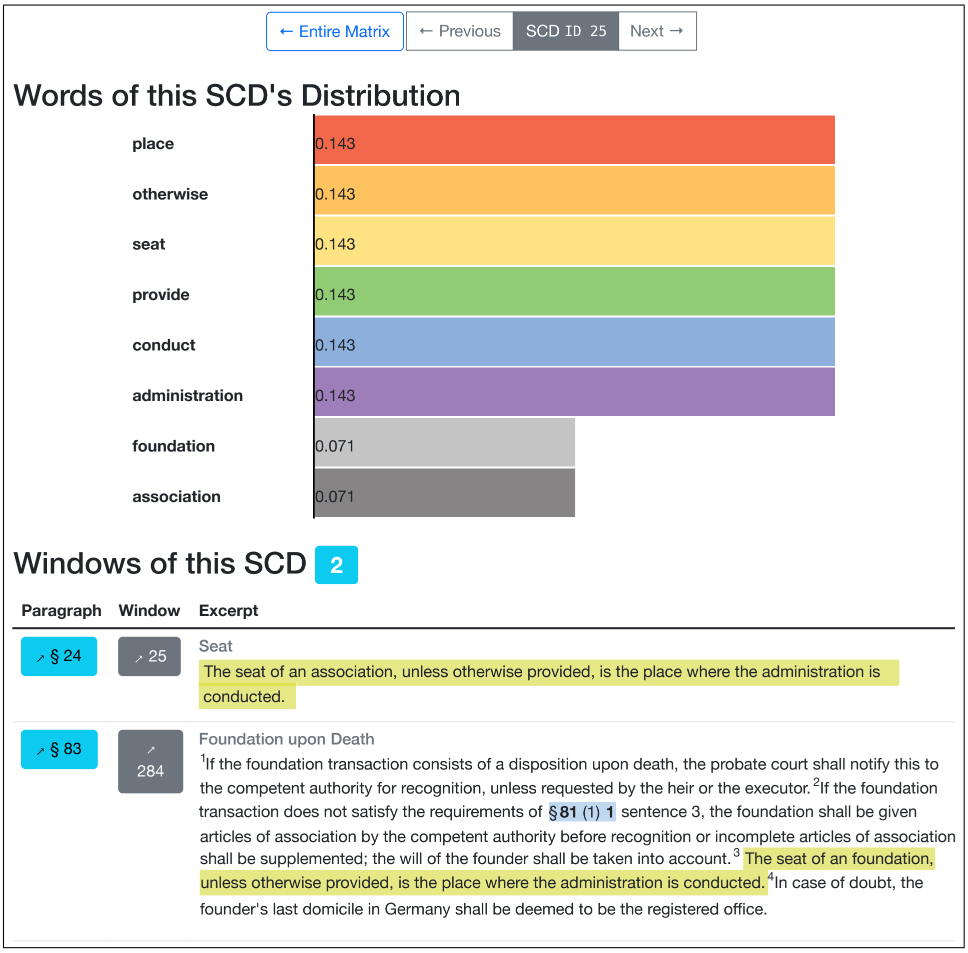

Figure 6. The SCD is visualized with its word-distribution and the referenced sentences. There is a list of referenced sentences, where each sentence is highlighted yellow. On the top it is possible to navigate through the SCDs of the corpus and view the entire SCD matrix.

browse all the SCDs estimated by UESM. The entire SCD matrix, the previous, and next SCD can be viewed with the upper buttons.

In Figure 7, the visualization of an estimated SCD matrix is shown. As there are many SCDs in one matrix, there are multiple pages with the SCDs on. A pagination on top allows to move from one page to the next. For each SCD the word-distribution, the ids of the referenced SCD windows, and the number of referenced windows are shown. It is possible to open the SCD with the blue button on the left and the referenced windows using the gray buttons on the right.

However, there is a huge amount of SCDs and *Charlie* can not go through all the

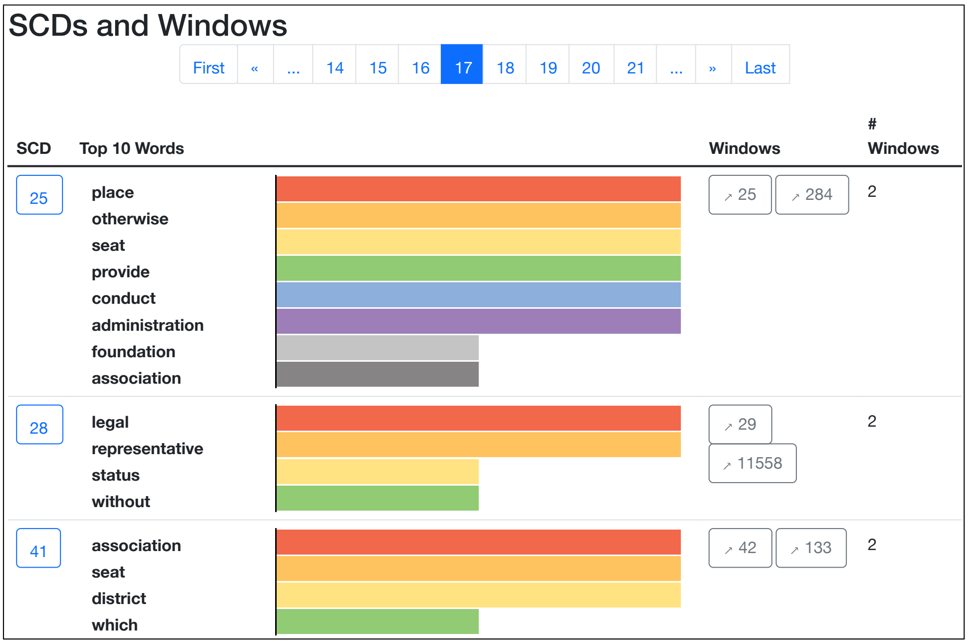

Figure 7. The SCDs of an SCD matrix are shown across multiple pages with an navigation between the pages on top. Each SCD is depicted by its word-distribution, the ids of the referenced SCD windows and the number of referenced windows. There is a button to view each SCD on the left side.

pages of SCDs to identify the most relevant SCD for *Charlie's* information need. Thus, the information system provides a MPS²CD based search through all the SCDs.

The MPS²CD algorithm [5] estimates a most probably suited SCDs for a single previously unseen text document. Here, the unseen text document is the user supplied query representing the user's information need. For this search query the most similar SCDs are calculated and shown to the user. Thus, the information system provides a powerful search using the SCDs of the corpus.

In Figure 8, the MPS²CD based search is shown. User *Charlie* types the query into the textbox on top and hits the search button. Then, the information system uses MPS²CD and shows a list of most probably suited SCDs in descending order of similarity. Again, for each SCD a list of referenced sentences is shown as excerpt of the corpus together with its surrounding content while the SCD window itself is highlighted yellow. Thus, *Charlie* can identify the best fitting SCD and view the SCD or the content by clicking the buttons.

Overall, the information system provides a human friendly way to interact with an information retrieval agent using UESM, while it runs unsupervised and with a small computational footprint. It allows users to import their corpora and browse

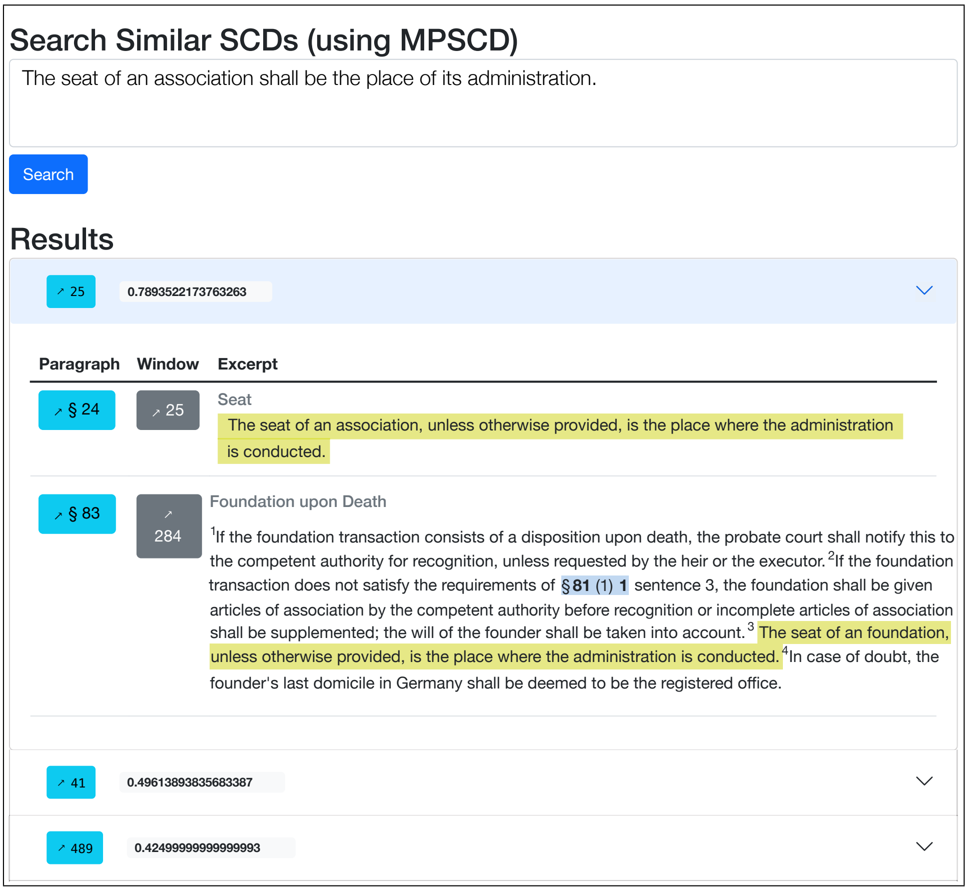

Figure 8. The MPS^{2}CD based search fetches the most similar SCDs based on a user supplied query. The query is inserted into the textarea on top and the similar SCDs are shown together with a similarity score and their referenced sentences.

all the documents while the estimated SCDs, representing the concepts and their locations across the corpus, are just a click away. Additionally, the MPS^{2}CD based search allows the users to formulate a query using their own words.

6. Related Work

Before we conclude, we take a look at related work. Adding data to corpora of text documents has been investigated for a long time [16]. Often the data associated with a corpus is denoted as an annotation.

In the beginning of natural language annotation, most annotations had to be added manually to the corpora. Even today, crowdsourcing can be used to manually annotate text documents [17]. Furthermore, semi-automatic and automatic annotation systems were developed, too, e.g., OpenIE [10]. Thus, unsupervised corpus annotation remains an important field of research.

In the context of SCDs, we interpret an SCD as a corpus annotation and in context of this paper, an SCD annotates multiple sentences of similar concepts all over the corpus' text documents. Topic models assign a distribution over the topics, estimated by the model itself, to each text document in the corpus and each topic is characterized by a distribution of occurring words in the topic's documents. Thus, similarly to the topics of a topic model, UESM associates text documents with SCDs representing concepts. A well-known topic modelling technique is LDA [4]. LDA is a generative model representing documents as a probability distribution over topics. Many extensions have been proposed to optimize the performance of LDA, e.g., the author-topic model [18], which extends LDA to couple each author of a document with a multinomial over words, and the dynamic topic model [19], which allows for analysing topic changes over time.

Documents assigned with a similar distribution over the topics, are assumed to be similar in terms of an topic model. However, LDA's perception of similar documents may not always match the human perception of similar documents [20].

UESM uses greedy similarity, K-Means or DBSCAN to identify similar sentences. Another technique to find similar sentences in a corpus of text documents is Similar Short Passages Identifier (SiSP) [21]. SiSP first extracts features from the sentences and then creates clusters of similar sentences. The authors evaluate the clusters found by SiSP against human labeled sentences. UESM may be used with SiSP, however, SiSP was developed for the Portuguese language.

A further idea is to cluster sentences hierarchically [22]. In difference to the clustering techniques used by UESM, the authors start with a sentence in the corpus and build a hierarchical clustering from this sentence. The hierarchical clustering has a tree-like structure, i.e., after staring from the first sentence, the tree branches across multiple levels to different concepts in the corpus.

Clustering can not only be used to identify similar sentences, it can also help to annotate sentences with their sentiment [23]. The authors assume, that two sentences in the same cluster have a similar sentiment and thus they can enrich the number of labels in a sparsely labeled corpus. In cases, where short sentences do not contain enough shared words to apply the cosine similarity, a ranking of the suitable clusters for each sentence can be used [24] to increase the performance of clustering techniques.

Text summarization is another field of research, where clusters of similar sentences are used. Thereby, the idea is to remove most of the similar sentences and only keep one sentence from each cluster. SimFinder [25] clusters small pieces of text, like sentences, into tight clusters. Unlike UESM, SimFinder does not work unsupervised, as it needs feature words in beforehand.

Another approach for text summarization is to extract the word vectors from each sentence and weight each word using tf-idf [26]. Then, the weighted word vectors are clustered using the cosine similarly with K-Means [2], again, from each cluster one sentences makes up the summary [27]. This approach overlaps with UESM in using word vectors, the cosine similarity, and K-Means. However, the authors pursue only the goal of text summarization, while UESM uses three methods and represents the concepts and topics of a corpus in the estimated SCD matrix.

7. Conclusion

This paper introduces UESM with three methods, namely K-Means, greedy similarity, and DBSCAN. UESM estimates SCD matrices for corpora of text documents in an unsupervised manner. Thereby, UESM detects sentences of similar concepts or topics in a corpus and then associates the same SCD to these similar sentences. Additionally, a model selection approach is introduced to detect the best method and hyperparameters of UESM for a corpus. Together with the model selection approach, UESM can can be integrated in an information system to assist humans working with corpora of text documents.

An SCD matrix for a corpus can be interpreted as a topic model of the corpus. Hence, the well-known LDA [4] is used to evaluate the performance of UESM. We use the UMass coherence to evaluate the quality of each model and show, that especially UESM using K-Means performs as good as LDA. In particular, LDA does not estimate an SCD matrix, but especially the SCD matrix is needed by approaches introduced by Kuhr et al. [5,6] and Bender et al. [7,9]. UESM enables the authors' approaches to be used in an unsupervised manner, i.e., without needing SCDs for a corpus in beforehand. Generally, without a focus on SCDs, UESM provides a new and powerful technique to create a topic model for a corpus. Overall, the evaluation shows that UESM using K-Means can keep up with LDA, while the greedy method is only slightly less powerful. Because DBSCAN associates too many sentences with the same SCD, DBSCAN is not suitable for most use-cases.

So far, we have introduced an SCD as a tuple of the SCD's additional data $\mathcal{C}$ and the referenced sentences. In this paper, we put efforts in finding the referenced sentences but we did not estimate the data $\mathcal{C}$. Thus, currently we only get the word probability distribution and the referenced sentences for each SCD without $\mathcal{C}$. Future work will focus on estimating $\mathcal{C}$, which is similar to automated topic naming for topic models [28]. Showing estimated names of the SCDs would be also a benefit for the users of the information system. Additionally, we will focus on using SCDs for information retrieval given requests from users, thus optimizing the estimated SCDs in a collaborative scenario [29].

Acknowledgment

The research of Marcel Gehrke was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2176 "Understanding Written Artefacts: Material, Interaction and Transmission in Manuscript Cultures", project no. 390893796. The research was conducted within the scope of the Centre for the Study of Manuscript Cultures (CSMC) at Universität Hamburg.

The authors thank the AI Lab Lübeck for providing the hardware used in the evaluation.

Bibliography

[1] F. Kuhr, T. Braun, M. Bender, and R. Möller, "To Extend or not to Extend? Context-specific Corpus Enrichment," *Proceedings of AI 2019: Advances in Artificial Intelligence*, pp. 357–368, 2019.

[2] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[3] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," pp. 226–231, 1996.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003. [Online]. Available: http://jmlr.org/papers/v3/blei03a.html

[5] F. Kuhr, M. Bender, T. Braun, and R. Möller, "Augmenting and automating corpus enrichment," *Int. J. Semantic Computing*, vol. 14, no. 2, pp. 173–197, 2020.

[6] ——, "Context-specific adaptation of subjective content descriptions," *15th IEEE International Conference on Semantic Computing, (ICSC 2021), Laguna Hills, CA, USA, January 27-29*, pp. 134–139, 2021.

[7] M. Bender, T. Braun, M. Gehrke, F. Kuhr, R. Möller, and S. Schiff, "Identifying and translating subjective content descriptions among texts," *Int. J. Semantic Computing*, vol. 15, no. 4, pp. 461–485, 2021.

[8] F. Kuhr, B. Witten, and R. Möller, "Corpus-Driven Annotation Enrichment," *13th IEEE International Conference on Semantic Computing, (ICSC 2019), Newport Beach, CA, USA, January 30 - February 1,*, pp. 138–141, 2019.

[9] M. Bender, F. Kuhr, and T. Braun, "To extend or not to extend? complementary documents," *16th IEEE International Conference on Semantic Computing, (ICSC 2022), Virtual, January 26-28*, pp. 17–24, 2022. [Online]. Available: https://doi.org/10.1109/ICSC52841.2022.00011

[10] G. Angeli, M. J. Johnson Premkumar, and C. D. Manning, "Leveraging linguistic structure for open domain information extraction," *Proceedings of the Association of Computational Linguistics (ACL)*, pp. 344–354, 2015.

[11] D. Pelleg, A. W. Moore *et al.*, "X-means: Extending k-means with efficient estimation of the number of clusters." in *Icml*, vol. 1, 2000, pp. 727–734.

[12] E. Hellinger, "Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen," *Journal für die reine und angewandte Mathematik*, pp. 210–271, 1909.

[13] M. Röder, A. Both, and A. Hinneburg, "Exploring the space of topic coherence measures," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, ser. WSDM '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 399–408.

[14] S. Vijayarani, J. Ilamathi, and S. Nithya, "Preprocessing techniques for text mining - an overview," *International Journal of Computer Science & Communication Networks*, vol. 5, p. 7–16, 2015.

[15] T. Sievers, M. Bender, and R. Möller, "Connecting AI technologies as online services to a humanoid service robot," in *15th International Conference on Computer and*

Automation Engineering (ICCAE). IEEE, 2023, pp. 431–435. [Online]. Available: http://dx.doi.org/10.1109/ICCAE56788.2023.10111181

[16] G. V. Maverick, "Computational analysis of present-day american english. henry kučera, w. nelson francis," *International Journal of American Linguistics*, vol. 35, no. 1, pp. 71–75, 1969. [Online]. Available: https://doi.org/10.1086/465045

[17] M. Sabou, K. Bontcheva, L. Derczynski, and A. Scharl, "Corpus annotation through crowdsourcing: Towards best practice guidelines," *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)*, 01 2014.

[18] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth, "The author-topic model for authors and documents," *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, p. 487–494, 2004.

[19] D. M. Blei and J. D. Lafferty, "Dynamic topic models," ser. ICML '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 113–120.

[20] W. B. Towne, C. P. Rosé, and J. D. Herbsleb, "Measuring similarity similarly: LDA and human perception," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 1, pp. 7:1–7:28, 2016. [Online]. Available: https://doi.org/10.1145/2890510

[21] E. R. M. Seno and M. d. G. V. Nunes, "Some experiments on clustering similar sentences of texts in portuguese," in *Computational Processing of the Portuguese Language*, A. Teixeira, V. L. S. de Lima, L. C. de Oliveira, and P. Quaresma, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 133–142.

[22] D. Kavyasrujana and B. C. Rao, "Hierarchical clustering for sentence extraction using cosine similarity measure," in *Emerging ICT for Bridging the Future - Proceedings of the 49th Annual Convention of the Computer Society of India (CSI) Volume 1*, S. C. Satapathy, A. Govardhan, K. S. Raju, and J. K. Mandal, Eds. Cham: Springer International Publishing, 2015, pp. 185–191.

[23] M. Hadano, K. Shimada, and T. Endo, "Aspect identification of sentiment sentences using a clustering algorithm," *Procedia - Social and Behavioral Sciences*, vol. 27, pp. 22–31, 2011, computational Linguistics and Related Fields. [Online]. Available: https://doi.org/10.1016/j.sbspro.2011.10.579

[24] L. Yang, X. Cai, Y. Zhang, and P. Shi, "Enhancing sentence-level clustering with ranking-based clustering framework for theme-based summarization," *Information Sciences*, vol. 260, pp. 37–50, 2014. [Online]. Available: https://doi.org/10.1016/j.ins.2013.11.026

[25] V. Hatzivassiloglou, J. L. Klavans, M. L. Holcombe, R. Barzilay, M.-Y. Kan, and K. McKeown, "SimFinder: A flexible clustering tool for summarization," 2001.

[26] J. E. Ramos, "Using tf-idf to determine word relevance in document queries," 2003.

[27] K. Shetty and J. S. Kallimani, "Automatic extractive text summarization using k-means clustering," in *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, 2017. [Online]. Available: https://doi.org/10.1109/ICEECCOT.2017.8284627

[28] A. Hindle, N. A. Ernst, M. W. Godfrey, and J. Mylopoulos, "Automated topic naming," *Empirical Software Engineering*, vol. 18, no. 6, pp. 1125–1155, 2013.

[29] K. Mahadik, Q. Wu, S. Li, and A. Sabne, "Fast distributed bandits for online recommendation systems," in *ICS '20: 2020 International Conference on Supercomputing, Barcelona Spain, June, 2020.* ACM, 2020, pp. 4:1–4:13.