Identifying Subjective Content Descriptions among Texts

Magnus Bender, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, and Simon Schiff
University of Lübeck
Institute of Information Systems
Ratzeburger Allee 160, 23562 Lübeck
{m.bender, braun, gehrke, kuhr, moeller, schiff}@ifis.uni-luebeck.de

***Abstract*—An agent pursuing a task may work with a corpus of documents as a reference library. Subjective content descriptions (SCDs) provide additional information that add value in the context of the task. On the pursuit of new documents to add to the corpus, an agent may come across new documents where content text and SCDs from another agent are interleaved and no distinction can be made unless the agent knows the content from somewhere else. Therefore, this paper presents an Hidden Markov model (HMM)-based approach to identifying SCDs based on SCD and word distributions in a previously unknown text where descriptions occur inline among content text. We present a case study evaluating the performance of identifying inline SCDs in a text based on real-world and simulated data.**

I. 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 fulfilling a defined task, e.g., providing document retrieval services given requests from users. We assume that the corpus represents the context of the task, since collecting documents 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 location-specific data as subjective content descriptions (SCDs). Generating a corpus specific SCD-word distribution, which encodes which words occur often with a given SCD, provides a value for different tasks of an agent in the context of a corpus, e.g., classifying new documents to extend a corpus with documents from a specific category [1] or enriching documents with SCDs associated to other documents in the same corpus [2].

So far, we have assumed that SCDs and documents are separate or at least clearly distinguishable. However, an agent in pursuit of new documents may come across documents where normal document text, i.e., content, and textual content descriptions, i.e., SCDs, are interleaved. An agent could go through the text and identify the SCDs manually or create a parser that separates the SCDs from the content, manually specifying rules for distinguishing content and SCDs. However, both approaches are cost-intensive and laborious, and require intimate knowledge about content and SCDs. A scenario we have encountered during a project features old poems in Tamil, which contain the poem itself and comments for specific words inline after the word. The poem would be the content and the comment fulfils the role of an SCD. Only an agent with knowledge of the original poem can distinguish poem and comment. Thus, the problem we tackle in this paper is that of identifying inline SCDs (iSCDs) automatically.

We turn to the agent's corpus of documents to solve the problem. Assuming that the unknown document with iSCDs is of the same context, we can use the corresponding SCDs-word distribution to distinguish between content and SCDs. The SCDs-word distribution allows for computing most probably suited SCDs (MPSCDs) for the unknown document based on the similarity between the words in the document and the words usually occurring with an SCD. If the words belong to the content, we expect that the agent is able to identify a corresponding MPSCD with a high similarity. If the words belong to an SCDs, which we assume has a different composition of words occurring together, we expect that the similarity value is low. Based on these expectations, we set up a hidden Markov model (HMM) where the hidden state variable encodes if the given words belong to content or SCDs and the observation sequence consists of discretized similarity values. Given this setup and the existing corpus, the agent can train the HMM and then compute a most-likely sequence of hidden states using the Viterbi algorithm [3]. Once iSCDs are identified, the agent can use them for whatever purpose, e.g., deciding if to incorporate the document into its corpus. Specifically, the contributions of this paper are:

(i) an approach to identify iSCDs based on an HMM trained on the available corpus of documents and SCDs and
(ii) a case study based on real-world (Tamil poems) and simulated data to evaluate the performance of our approach.

The remainder of this paper is structured as follows: We first specify notations and recap SCD-word distributions. Then, we present our HMM-based approach to identify iSCDs in a new document and evaluate the performance in a case study. The case study is followed by a look at related work. Last, we conclude and present future work.

II. Preliminaries

This section specifies notations and recaps the concept of an SCD-word distribution.

A. Subjective Content Descriptions

We define the following terms to formalize the setting of a corpus with documents and SCDs.

- A word w is a basic unit of discrete data from a given vocabulary $\mathcal{V} = (w_1, \ldots, w_N)$, $N \in \mathbb{N}$, and can be represented a word w as a one-hot vector of length N having a value of 1 where $w = w_i$ and 0's otherwise.
- A document d is represented by a sequence of $D \in \mathbb{N}$ words $(w_1^d, \ldots, w_D^d)$, where each word $w_i^d \in d$ is a subset of vocabulary $\mathcal{V}$. The function $\#word(d)$ returns the total number of words occurring in document d.
- A corpus $\mathcal{D}$ represents a set of $Z \in \mathbb{N}$ documents $\{d_1, \ldots, d_Z\}$ and we assume that the documents are from the same context. The term $\mathcal{V}_\mathcal{D}$ refers to the corpus-specific vocabulary representing the set of all words occurring in the documents of corpus $\mathcal{D}$.
- A SCD t is a sequence of words $(w_1^d, \ldots, w_M^d)$, $M \in \mathbb{N}$, where each word $w_i^d \in t$ is a subset of vocabulary $\mathcal{V}_\mathcal{D}$, and t can be associated with a position ρ in a document d. We use the term located SCDs interchangeably for associated SCDs and represent a located SCD t by the tuple $\{(t, \{\rho_i\})_{i=1}^l\}$, where $\{\rho_i\}_{i=1}^l$ represents the $l \in \mathbb{N}$ positions in document d the SCD t is associated with.
- For each document $d \in \mathcal{D}$ there exists a set g denoted as SCD set containing a set of m located SCDs $\{t_j, \{\rho_i\}_{i=1}^{l_j}\}_{j=1}^m$. Given a document d or a set g, the terms $g(d)$ and $d(g)$ refer to the set of located SCDs in document d and the corresponding document d, respectively. The set of all located SCD tuples in corpus $\mathcal{D}$ is then given by $g(\mathcal{D}) = \bigcup_{d \in \mathcal{D}} g(d)$.
- For each located SCD t_j in $g(d)$ exists an corresponding SCD window $win_{d,\rho}$ that refers to a sequence of words in d surrounding ρ in d, i.e., $win_{d,\rho} = (w_{(\rho-i)}^d, \ldots, w_\rho^d, \ldots, w_{\rho+i}^d)$, $i \in \mathbb{N}$ and ρ marks the middle of the window. The window-specific position of $w^d \in win_{d,\rho}$ is given by $pos(w^d, win_{d,\rho})$ (0-based numbering) and the size of $win_{d,\rho}$ is given by $s(win_{d,\rho}) = 2i + 1$.
- Each word $w^d \in win_{d,\rho}$ is associated with an influence value $I(w^d, win_{d,\rho})$ representing the distance in the text between w^d and position ρ. The closer w^d is positioned to ρ in $win_{d,\rho}$, the higher its corresponding influence value $I(w^d, win_{d,\rho})$. The influence value of w^d at $pos(w^d, wind_{d,\rho})$ is distributed binomial, i.e., $I(w^d, win_{d,\rho}) = \binom{n}{k} \cdot q^k \cdot (1-q)^{n-k}$, where $n = s(win_{d',\rho})$, $k = pos(w^d, win_{d,\rho})$, and $q = \frac{\rho}{n}$.

B. SCD-word Distribution

An SCD-word distribution encodes which words occur often around the position of a given SCD. Specifically, we generate an additional representation for each of the m SCDs associated to documents in corpus $\mathcal{D}$ by building a vector of length n, where $n = |\mathcal{V}_\mathcal{D}|$, s.t. each vector entry refers to a word w in $\mathcal{V}_\mathcal{D}$. The entry itself is a probability describing how likely it is that a word occurs in an SCD window surrounding the position associated with the SCD, yielding an SCD-word distribution for each SCD. Algorithm 1 generates the SCD-word distribution for all m SCDs available in SCD set $g(\mathcal{D})$. We represent the SCD-word distribution by an $m \times n$ matrix $\delta(\mathcal{D})$, where the SCD-word distribution vectors form the rows of the matrix:

$$\delta(\mathcal{D}) = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_m \end{array} \begin{array}{c} \begin{array}{ccccc} w_1 & w_2 & w_3 & \cdots & w_n \end{array} \\ \begin{pmatrix} v_{1,1} & v_{1,2} & v_{1,3} & \cdots & v_{1,n} \\ v_{2,1} & v_{2,2} & v_{2,3} & \cdots & v_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{m,1} & v_{m,2} & v_{m,3} & \cdots & v_{m,n} \end{pmatrix} \end{array} \quad (1)$$

Algorithm 1 Forming SCD-word distribution matrix $\delta(\mathcal{D})$

```
1:  function BUILDMATRIX(Corpus D)
2:      Input: D
3:      Output: δ(D)
4:      Initialize an m × n matrix δ(D) with zeros
5:      for each d ∈ D do
6:          for each t, ρ ∈ g(d) do
7:              for each w ∈ win_{d,ρ} do        ▷ Iterates over ρ
8:                  δ(D)[t][w] += I(w, win_{d,ρ})
9:          Normalize δ(D)[t]
10:     return δ(D)
```

The input of Alg. 1 is a corpus $\mathcal{D}$ containing a set of documents associated with SCDs. In line 4 of Alg. 1, we instantiate an empty $\delta(\mathcal{D})$ by filling the matrix with zeros. Afterwards, we update the SCD-word distribution matrix $\delta(\mathcal{D})$ entries based on the SCDs and words occurring in the documents of $\mathcal{D}$ using maximum-likelihood estimation by counting, for each SCD t, the number of occurrences of each word w in the corresponding windows $win_{d,\rho}$ of all documents in $\mathcal{D}$ and all positions. We weight the occurrences by the influence value of each word in a window (line 8). At the end of the outer loop, the SCD-word distribution of the current SCD t is normalized to yield a probability distribution for each SCD over the complete vocabulary (line 9). Finally, Alg. 1 returns the SCD-word distribution matrix $\delta(\mathcal{D})$.

Given $\delta(\mathcal{D})$ and presented with a new document d of the same context containing pure text without any SCDs, an agent can use $\delta(\mathcal{D})$ to find M MPSCDs. MPSCDs are computed by sliding a toppling window $win_{d,\rho}$ of size $\sigma = \frac{\#word(d)}{M}$ over the words in d and computing the similarity between the words in $win_{d,\rho}$ weighted according to $I(w^d, win_{d,\rho})$ and the rows in $\delta(\mathcal{D})$ and choose the SCD t belonging to the row with highest similarity. Afterwards, width and position of each window can be further optimized. Please refer to [1] for details.

Next, we present our approach to identifying SCDs in an unknown document with iSCDs.

III. Identifying Inline SCDs

This section introduces the problem of identifying textual SCDs that are interleaved with content text in a document and presents an HMM-based approach solving the problem.

A. Inline SCD Problem

The problem at hand consists of an agent being faced with a document containing content in the form of text and textual SCDs and no markers or way inherently available to distinguish the two. An important task of an agent is to identify iSCDs among text s.t. the agent can (i) reconstruct the content of a document containing iSCDs, and (ii) use the located iSCDs, e.g., to identify similar documents within the corpus. We refer to SCDs that are interleaved with the text of a document as *iSCDs* and to the remaining text as *content*. Given this new setting, we introduce iSCDs into our notation.

- An iSCD t is a sequence of words $(s_1, \ldots, s_n), n \in \mathbb{N}, s_i \in \mathcal{V}_{g(\mathcal{D})}$ that is associated to the sequence of words exactly preceding t in d.
- Next to the vocabulary $\mathcal{V}_{\mathcal{D}}$ of corpus $\mathcal{D}$, there is a vocabulary $\mathcal{V}_{g(\mathcal{D})}$ of the words occurring in the (inline) SCDs $t \in g(\mathcal{D})$. The two vocabularies may overlap.
- A document d is a sequence of words from $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{g(\mathcal{D})}$ with subsequences of words from $\mathcal{V}_{\mathcal{D}}$ and subsequences of words from $\mathcal{V}_{g(\mathcal{D})}$ alternating, where the latter is associated with the preceding window of words. Further SCDs may be located throughout d.

Problem 1 introduces the inline SCD problem and Example 1 illustrates the problem using a short text.

Problem 1 (Inline SCD Problem)**.** *An agent does not know which subsequences of words are content and which are iSCDs for a document* $d = (w_1^d, \ldots, w_D^d), w_i^d \in (\mathcal{V}_{\mathcal{D}} \cup \mathcal{V}_{g(\mathcal{D})})$.

Example 1 (Inline SCD Example)**.** *Assume that a new document* $d \notin \mathcal{D}$ *contains the following sentence with two iSCDs:*

> *"David Blei professor at Columbia University received the ACM Infosys Foundation Award renamed in the ACM Prize in Computing in 2013."*

The two iSCDs are underlined. However, the highlighting is not available in the original document. An agent is faced only with the word sequence and has to decide which words represent iSCDs. The problem is simple if the two vocabularies do not overlap but becomes increasingly harder the more the two vocabularies share words and those words have similar frequencies regarding occurrences.

We can formulate the inline SCD problem as a classification problem estimating for each word in a sequence of words the corresponding category. In our setting, we have two categories. One category represents the subsequences in d that are not part of an iSCD, i.e., content, and another category represents the subsequences in d belonging to an iSCD.

B. General Procedure

To solve the problem, we work with two assumptions about corpus $\mathcal{D}$ and document d carrying Problem 1:

(i) Document d belongs to the same context as $\mathcal{D}$.
(ii) Vocabulary $\mathcal{V}_{\mathcal{D}}$ or the words occurring together in a window of an associated SCD differ from vocabulary $\mathcal{V}_{g(\mathcal{D})}$ or the words occurring together in the SCD.

Algorithm 2 Estimating MPSCD sequence

function ESTIMATEMPSCDSEQUENCE(d, σ, $\delta(\mathcal{D})$)
  $\rho \leftarrow \frac{\sigma}{2}$, $\mathcal{W} \leftarrow \emptyset$
  for $\rho \leftarrow \frac{\sigma}{2}$; $\rho \leq words(d)$; $\rho \leftarrow \rho + 1$ **do**
    Set up $win_{d,t,\rho}$ of size σ around ρ with $t = \perp$
    $\delta(win_{d,t,\rho}) \leftarrow$ new zero-vector of length n
    for $w \in win_{d,t,\rho}$ **do**
      $\delta(win_{d,t,\rho})[w]$ += $I(w, win_{d,t,\rho})$
    $t \leftarrow \arg\max_{t_i} \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d,t,\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d,t,\rho})|}$ in $win_{d,t,\rho}$
    $sim \leftarrow \max_{t_i} \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d,t,\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d,t,\rho})|}$
    $\mathcal{W} \leftarrow \mathcal{W} \oplus (sim, win_{d,t,\rho})$
    $g(d) \leftarrow g(d) \oplus t$
  return $g(d)$, $\mathcal{W}$

Given the first assumption, we can use the SCD-word distribution $\delta(\mathcal{D})$, generated based on $\mathcal{D}$, for d as well. Given the second assumption, $\delta(\mathcal{D})$ will work well to estimate MPSCDs for content words but less well for iSCDs words. If we estimate MPSCDs for d but using a sliding window (instead of a tumbling one), we expect the following behavior: The MPSCD similarity value for a window containing an iSCD should be significantly smaller than the MPSCD similarity value for a window containing content of d related to the content of other documents in $\mathcal{D}$. Therefore, if sliding a window over the words in d, we expect the similarity values to show a characteristic behavior, which we can train an HMM with. Specifically, one has to perform the following steps to solve Problem 1:

1) Estimate $\delta(\mathcal{D})$ for $\mathcal{D}$ using Alg. 1 (offline).
2) Train an HMM for classifying whether a word belongs to the category "content" or the category "iSCD" (offline).
3) For each new document d carrying Problem 1:
   a) Estimate the MPSCD sequence over a sliding window along the words of d using $\delta(\mathcal{D})$.
   b) Compute the most probable sequence of states in the HMM given the sequence of similarity values of the MPSCD sequence as evidence.
   c) From the most probable sequence of states, identify the words in d that belong to category "iSCD".

The following sections present in detail how to estimate the MPSCD sequence and classify iSCDs using an HMM.

C. Estimating an MPSCD Sequence

Given a corpus $\mathcal{D}$ containing documents associated with a set of location-specific SCDs, we can generate a corpus-specific SCD-word distribution $\delta(\mathcal{D})$ using Alg. 1. Based on $\delta(\mathcal{D})$, Alg. 2 allows for calculating a sequence of MPSCDs similarity values for a document d by sliding a window $win_{d,t,\rho}$ of size σ over the words in d. Initially, the sliding window contains the first σ words $(w_1^d, ..., w_\sigma^d)$. Then, we shift the window over the sequence of words in d by removing the first word w_1^d and extending the window with the word $w_{\sigma+1}^d$ an so forth until the last window contains $(w_{(D-\sigma)}^d, ...w_D^d)$. Example 2 describes the sliding window behaviour in detail.

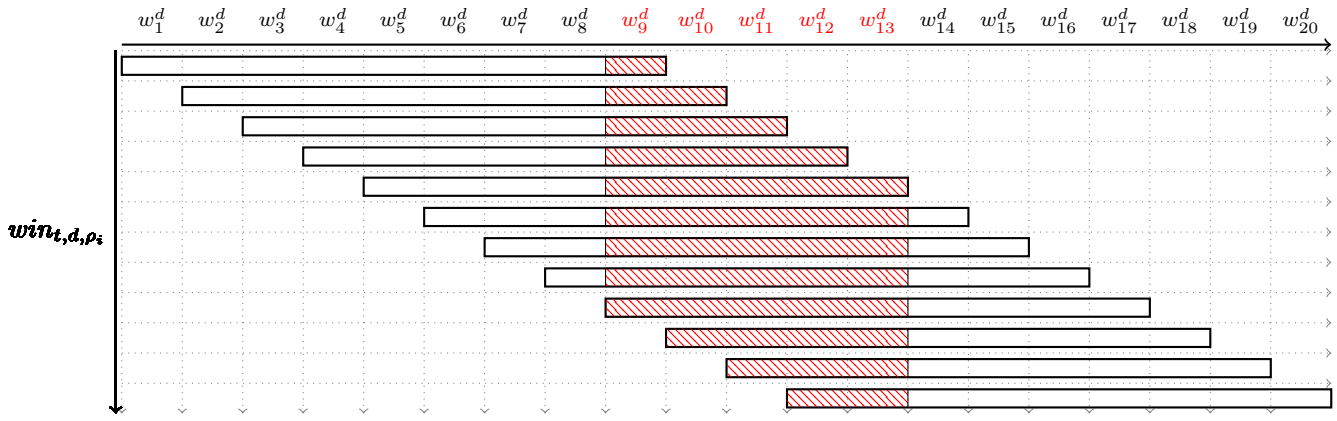

Fig. 1: Window of size $\sigma = 9$ sliding over the words in document d where words w_9^d to w_{13}^d represent an inline SCD.

Example 2 (Sliding Window)**.** *Figure 1 illustrates the behavior of a sliding window $win_{d,t,\rho}$ for the first 20 words w_1^d to w_{20}^d in d with a window size σ of 9, starting with $win_{d,t_1,w_5^d} = (w_1^d, ..., w_9^d)$ and ending with $win_{d,t_{12},w_{16}^d} = (w_{12}^d, ..., w_{20}^d)$. The words w_9 to w_{13} represent an iSCD.*

For each sequence of words in $win_{d,t,\rho}$, Alg. 2 estimates the MPSCD t and corresponding similarity value sim. Specifically, Alg. 2 builds a vector representation $\delta(win_{d,\rho})$ for the sequence of words in $win_{d,t,\rho}$ and compares the vector with the vector representations of all SCDs in $\delta(\mathcal{D})$. Finally, Alg. 2 uses SCD t from $\delta(\mathcal{D})$ that has a vector representation most similar to $\delta(win_{d,\rho})$ as MPSCDs. Similarity is defined using the cosine similarity. Example 3 describes what similarity values might look like given the sliding window of Example 2.

Example 3 (Sequence of Similarity Values)**.** *Figure 3 represents the sequence of MPSCD similarity values for the sliding window used in Fig. 1. We have a sequence of 12 similarity values for the words in the corresponding 12 windows. In the first window win_{d,t_1,w_5^d}, only the word w_9 belongs to an iSCD. Shifting the window to the right results in a second word belonging to the iSCD. The more words of a window belong to an iSCD, the lower the corresponding similarity value.*

As shown in Fig. 3, iSCDs yield a specific pattern in the sequence of similarity values. Next, we present how to identify iSCDs in d using an HMM given this sequence.

D. Estimating iSCDs

We use an HMM to estimate the iSCDs using the sequence of similarity values from the MPSCDs in d. We first define the HMM and then present how to estimate the iSCDs.

Definition 1 (Hidden Markov model)**.** *A hidden Markov model $\lambda = (a_{ij}, b_j, \pi)$ for solving Problem 1 is defined by:*

- *(hidden) states given by $\Omega = \{s_1, ..., s_n\}$, where $n = 2$, with state s_1 meaning the word behind s_1 belongs to "content" and s_2 meaning the word belongs to "iSCD",*
- *an observation alphabet $\Delta = \{y_1, \ldots, y_m\}$, where each y_i represents a range of MPSCD similarity values,*

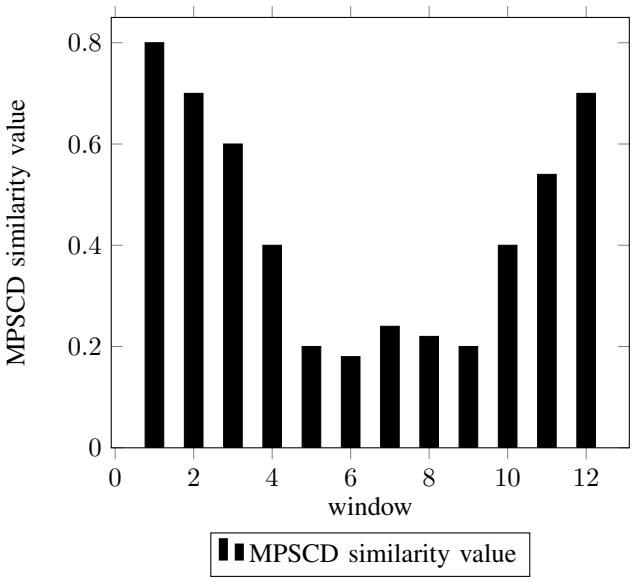

Fig. 3: Sequence of MPSCD similarity values corresponding to the word sequences from the 12 windows in Figure 1.

- *a transition probability matrix A representing the probability between all possible state transitions $a_{i,j}$ between the two states $s_1, s_2 \in \Omega$.*
- *an emission probability matrix B representing the probability to emit a symbol from observation alphabet Δ for each possible state in Ω, and*
- *an initial state distribution vector $\pi = \pi_0$.*

With $\sum_{j=1}^{n} a_{i,j} = 1$, the entries of transition probability matrix A between states $s_i, s_j \in \Omega$, are given by

$$a_{i,j} = P(s_j|s_i).$$

The entries of emission probability matrix B represent the probability to emit symbol $y_k \in \Delta$ in state $s_j \in \Omega$ and, with $\sum_{j=1}^{m} b_j(y_k) = 1$, are given by

$$b_j(y_k) = P(y_k|s_j).$$

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

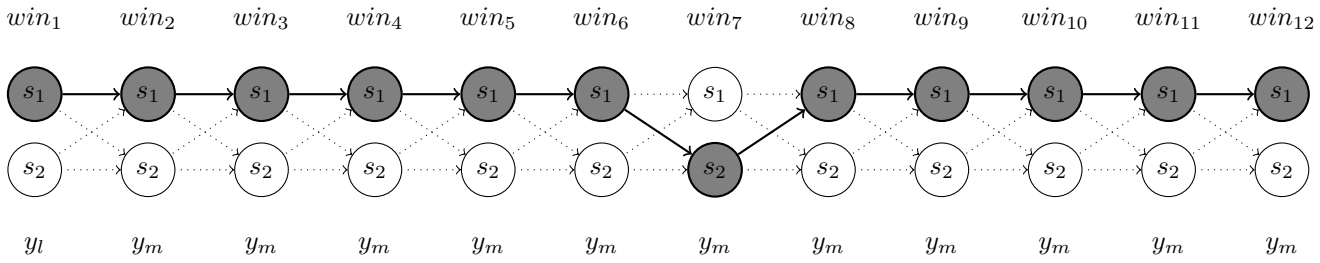

Fig. 2: Trellis corresponding to the MPSCD sequence resulting from sliding windows in Example 2.

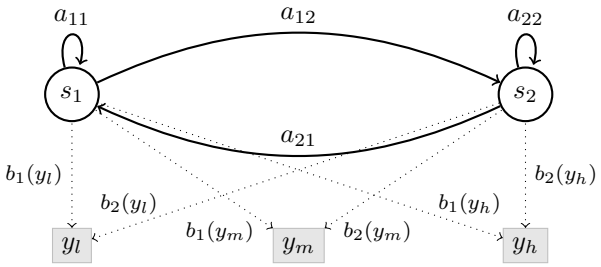

Fig. 4: Hidden Markov model containing two hidden states $\{s_1, s_2\}$ emitting three observation symbols $\{y_l, y_m, y_h\}$.

Example 4 (Graphical Representation)**.** *Figure 4 contains a graphical representation of a hidden Markov model* $\lambda = (a_{ij}, b_j, \pi)$ *containing observation symbols* $\{y_l, y_m, y_h\} \in \Delta$*, and the two states* $\{s_1, s_2\} \in \Omega$*.*

Generally, A and B are unknown and have to be learned, e.g., using the Baum-Welch algorithm [4]. Using a set of documents containing located SCDs, we can calculate MPSCDs and their similarity values to train an HMM on this information using the Baum-Welch algorithm. The discrete observation alphabet Δ requires discretizing similarity values. A discretization function $f : [0, 1] \mapsto \Delta$ maps a similarity value x to one of the m symbols in Δ. Algorithm 3 shows pseudocode, mapping a sequence of similarity values $\mathcal{W}$ to the observation symbols in Δ based on f, returning a sequence of observable symbols O. Generally, the discretization depends on the task of an agent and can be adapted to each problem individually.

To solve the iSCD problem, we have to find the most likely sequence of states from alphabet Ω, given a sequence of observation symbols from alphabet Δ. Given a trained HMM and the discretization procedure in Alg. 3, the workflow for estimating iSCDs based on a sequence of similarity values $\mathcal{W}$ over the windows in document d is as follows:

1) Discretize the similarity values in $\mathcal{W}$, leading to an observation sequence O.
2) Based on O, compute the most likely state sequence S.
3) Based on S, extract the windows win_i from states in S where $s = s_2$.
4) For each extracted window win_i, reconstruct the original window $win_{d,\rho}$ by setting $\rho = i + \lfloor \frac{\sigma}{2} \rfloor$. Mark the words in $win_{d,\rho}$ as the iSCD.

Algorithm 3 MPSCD Similarity Discretization

```
function DISCRETIZE(W, f)
    O ← ()                                   ▷ observation sequence
    for each sim ∈ W do
        y_k ← f(sim)                         ▷ y_k ∈ Δ
        O ← O ∘ y_k
    return O
```

We calculate the most likely sequence of states using the Viterbi algorithm [3], which makes use of the dynamic programming trellis for computing the most likely state sequence S for an observation sequence O. Let us consider an example of a most likely sequence given the sequence of MPSCD similarity values resulting from the 12 windows in Example 2.

Example 5. *Assume that Alg. 2 yields the following similarity values for the 12 windows in Example 2, as shown in Fig. 3:*

$$(0.8, 0.7, 0.6, 0.4, 0.2, 0.18, 0.24, 0.22, 0.22, 0.4, 0.54, 0.7)$$

Using the following function for discretization

$$f(x) = \begin{cases} y_l & 0 \leq x < th_1 \\ y_m & th_1 \leq x < th_2 \\ y_h & th_4 \leq x \leq 1, \end{cases}$$

where $th_1 = 0.3$*, and* $th_2 = 0.7$*, we get the observation sequence:*

$$O = (y_h, y_h, y_m, y_m, y_l, y_l, y_l, y_l, y_l, y_m, y_m, y_h)$$

Let the corresponding state sequence in the trained HMM be given by

$$S = (s_1, s_1, s_1, s_1, s_1, s_1, s_2, s_1, s_1, s_1, s_1, s_1).$$

Figure 2 represents the trellis of the observation sequence O, where the thick arrows indicate the most probable transitions between the states and the dotted lines represent all possible state transitions. The hidden iSCDs is at position win_7*, which corresponds to window* $win_{d,\rho}$ *with* $\rho = 7 + \lfloor \frac{9}{2} \rfloor = 11$*. The identified window contains the words* $(w^d_9, \ldots w^9_{13})$*, which make up the iSCD in the example.*

Correctness By identifying the M word sequences that most probably are iSCDs, we automatically identify which words belong to iSCDs and which belong to content, i.e.,

$d \setminus \{(s_1^m, \ldots, s_{n_m}^m) \mid s_i \in d, m \in \{1, \ldots, M\}\}$. Therefore, we solve Problem 1 by not only providing which subsequences are iSCDs but providing those that are most probable given the underlying HMM, which makes the quality of the solution to a specific instance of the problem optimal in the sense that given the probabilistic fundamentals of our approach, this calculated solution is the one leading to the highest probability.

Next, we present a case study evaluating our approach to solving the iSCDs problem.

IV. Case Study

After introducing the HMM-based approach to detect iSCDs in documents, we present a case study analyzing the performance of estimating iSCDs within documents of different data sets. We describe the data sets, evaluation workflow, and present the results.

A. Data Sets

We use three data sets to evaluate the performance of the HMM-based approach to identify iSCDs in a new document.

1) Data set `Tamil` consists of 91 poems that are transcribed from approximately three-hundred-year-old palm leaves [5].
2) Data set `US` consists of 74 articles about cities in the Unites States of America.[1]
3) Data set `EU` consists of 10 articles about cities in Europe.[2]

For the first data set, we extract the documents (poems) from [5]. Each document is associated with iSCDs representing comments about the original text of the corresponding document. The second and third data set consist of articles from the open and widely accessible online encyclopedia *Wikipedia*. We download all articles using a Python script and the Wikipedia TextExtracts API. Afterwards, we store the documents in the respective corpora and preprocess the documents by performing following tasks: (i) lowercasing all characters, (ii) stemming the words, (iii) tokenizing the result, and (iv) eliminating tokens from a stop-word list containing 337 words. These four tasks are standard preprocessing tasks in the NLP community transforming the text of documents into more digestible form for machine learning algorithms to increase their performance [6]. Documents in the first data set contain text from the 17th and 18th century. As language changes over time, we cannot apply modern word stemmers and stopwords for the Tamil language to the first data set.

However, articles published on Wikipedia do not contain iSCDs. Thus, we generate iSCDs for each document of the second and third data set using the free online dictionary *Wiktionary* by (i) downloading a dump of the English Wiktionary, (ii) creating an iSCD for each word in a document using the corresponding Wiktionary entry (if available), and (iii) adding the iSCDs to the document.

Table I gives an overview about the average length of documents, size of the vocabularies, and iSCDs etc.

[1]US cities – https://bit.ly/3jUua5H

[2]European cities – https://bit.ly/34WXMsE

TABLE I: Characteristics of the three data sets

	Tamil	US	EU
# Documents	91	74	10
Avg $\#word(d)$	73.2	200.7	318.7
# iSCDs	997	1814	757
# Vocabulary $\mathcal{V}_\mathcal{D} \cup \mathcal{V}_{g(\mathcal{D})}$	8031	6688	3314
# Vocabulary $\mathcal{V}_\mathcal{D}$	5521	3657	1439
# Vocabulary $\mathcal{V}_{g(\mathcal{D})}$	2684	3902	2232
# Vocabulary $\mathcal{V}_\mathcal{D} \cap \mathcal{V}_{g(\mathcal{D})}$	174	871	357

B. Evaluation Workflow

We evaluate the performance of the HMM-based approach to identify SCDs among text. The HMM contains the two hidden states (s_1,s_2) as defined in Def. 1 and five observable states y_1 to y_5 representing the following similarity intervals, y_1 = [0.0,0.1), y_2 = [0.1,0.2), y_3 = [0.2,0.4), y_4 = [0.4,0.7), and y_5 = [0.7,1.0], with a corresponding discretization function. We have tested different setups and five observable symbols and the specific intervals have shown to be a good setup for all three data sets. Then, we perform the following five tasks for each data set:

(i) Generate a training set containing 90% of documents and a test set with the remaining 10% of documents.
(ii) Form the SCD-word distribution δ for the training set using Alg. 1.
(iii) Generate an HMM and apply the Baum-Welch algorithm [3] to train the parameters of the HMM.
(iv) For each document in the test set:
  (a) Estimate MPSCDs and the corresponding MPSCD similarity values using Alg. 2.
  (b) Compute the most probable sequence of hidden states in the HMM for the discretized sequence of similarity values using the Viterbi algorithm.

We evaluate the performance of the HMM-based approach by comparing the results with the following two standard approaches, namely, word-based classification and a single threshold-based classification.

Word-based Classification For each word, we form a distribution representing how often the word occurs in content ($\#_c$) vs. iSCD ($\#_s$), i.e., $p = \frac{\#_c}{\#_c + \#_s}$ and $1 - p$. We classify each word by sampling from $(p, 1-p)$. Words belonging only to one vocabulary have a $(1, 0)$-distribution and can be directly classified as belonging to either content or iSCD. For words that are not part of any vocabulary, we randomly assign a category.

Threshold-based Classification Instead of training an HMM on the MPSCD similarity sequences, we directly classify based on the MPSCD similarity value of a window sim and a threshold ℓ. If $sim < \ell$, we classify the words in the window as an iSCD. For the first data set, $\ell = 0.1$ results in best iSCD detection performance. For the second and third data set, $\ell = 0.3$ leads to best results.

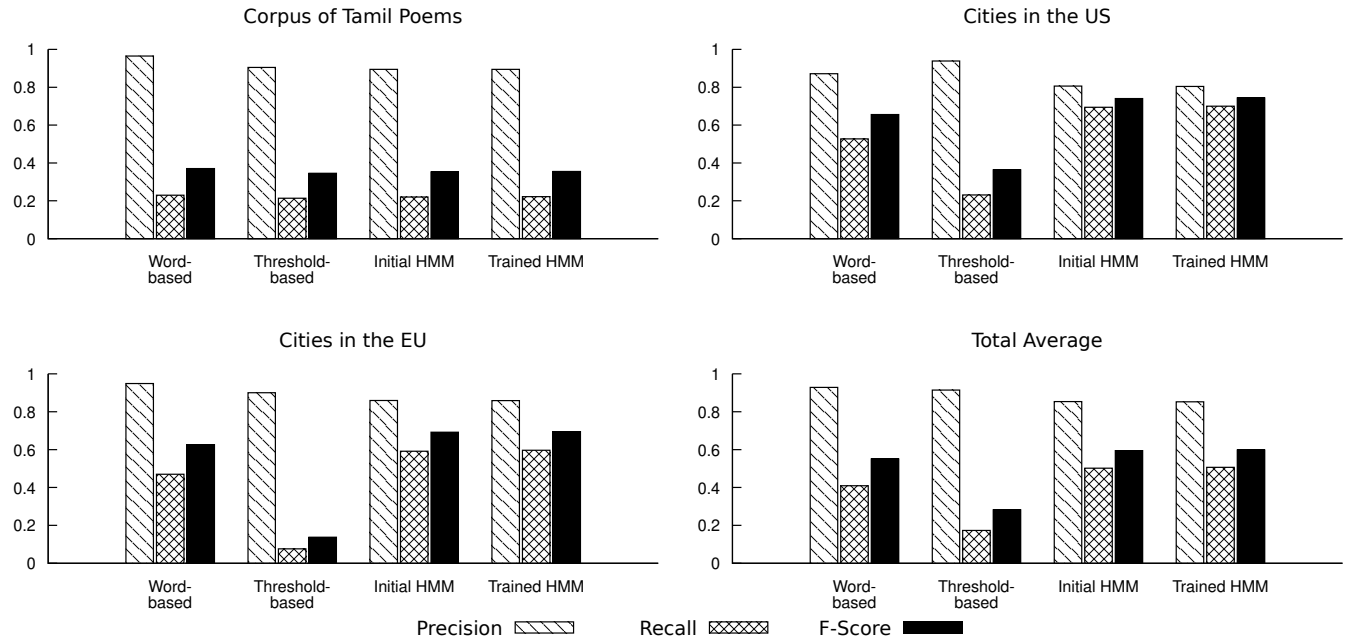

Fig. 5: Performance of the HMM-based approach compared to the word-based and threshold-based approach for iSCD classification.

C. Results

We evaluate the performance of the HMM-based approach to identify SCDs among texts and compare the performance with the word-based and threshold-based classification approaches. We use the F1-score to evaluate the performance of each approach, which is defined by:

$$\text{F1-Score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}},$$

where precision and recall are defined by:

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \text{Recall} = \frac{tp}{tp + fn},$$

with tp = # true positives, fp = # false positives, tn = # true negatives, and fn = # false negatives. We use cross validation with 90% of the documents for training and 10% of the documents for testing, resulting in the total average of the precision, recall and F1-score.

As depicted in Table I, the documents in `Tamil` contain only 174 words, representing 2% of all words, that occur in both $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{g(\mathcal{D})}$. The `US` and `EU` set contain 871 (13.0%) and 357 (10.7%) words occurring in both vocabularies, respectively. Thus, we expect a good word-based classification performance for the `Tamil` set and a good performance by the HMM-based approach for the `US` and `EU` set.

Figure 5 presents the performance (precision, recall, F1-score) of the word-based, threshold-based, and HMM-based approach for all three data sets. For the HHM-based approach, we present the performance of the initial model and the trained model. The initial HMM contains the following emission probabilities in case of iSCDs:

$$\{y_1 : 0.50, y_2 : 0.35, y_3 : 0.15, y_4 : 0.00, y_5 : 0.00\}$$

and in case of text:

$$\{y_1 : 0.00, y_2 : 0.10, y_3 : 0.50, y_4 : 0.35, y_5 : 0.05\}.$$

The emission probabilities encode that a window associated with an MPSCD of a high similarity is unlikely being classified as an iSCD. We train this initial HMM, using 90% of the data for training and 10% for testing, with the Baum-Welch algorithm. The result is the trained model referenced in Fig. 5.

The word-based classification yields best results for the `Tamil` set. The recall of the word-based classification for `US` and `EU` is considerably smaller than the recall of the HMM-based approach leading to a smaller F1 score. Interestingly, the word-based classification performance is not as poor as expected for data sets containing a more overlapping vocabulary between content and SCDs.

The threshold-based classification performance is worst for all three data sets. The approach identifies only a small number of iSCDs with many false negatives. One reason for the large number of false-negative iSCD classification is that all words in a window are classified into the same category. Even if testing with different thresholds, the recall remains low.

The HMM-based approach performs well for all three corpora. The trained HMM performs slightly better than the initial HMM for all three corpora. The difference between the initial and the trained model is not as pronounced since the initial values for the emission probabilities are already of good quality, which reduces not only runtimes for learning but also has the effect of the initial model performing relatively well.

Overall, the HMM-based approach yields the best performance in the case study, outperforming the word-based and threshold-based approach in terms of recall and F1-score.

V. Related Work

We look at related work in the area of text segmentation, where the goal is to distinguish words belonging to different categories as well, as well as HMM-based classification.

In our setting, we need to separate words that are part of the content of a document from words that are part of an iSCD. In text segmentation, the task is to separate text into segments, such as words [7], topics [8], sentences [9], passages, or lines. The difficulty lies in identifying the segment borders [10]. Choi [11] constructs a dictionary of word stem frequencies for each sentence and represents it as a vector of frequency counts for domain independent linear text segmentation. A similarity matrix is constructed by comparing all sentences using the cosine similarity. The values in the matrix are then replaced by the number of neighbouring elements with a lower similarity. Segments then are identified as a square region along the diagonal of the rank matrix.

The existing algorithms do not solve the iSCD problem as they ignore the context of a specific task, which we explicitly incorporate. In addition, we identify exactly those segments as iSCDs that are relevant for the task of an agent.

Another class of related work deals with HMM-based classification. Classification and statistical learning using HMMs has achieved remarkable progress in the past decade. Using a HMM is a well-researched stochastic approach for modeling sequential data, and it has been successfully applied in a variety of fields, such as speech recognition [12], character recognition [13], finance data prediction [14], [15], credit card fraud detection [16], and workflow mining [17], [18], [19]. With the identification of iSCDs, we have successfully applied HMMs in another context, solving a new task.

VI. Conclusion

This paper presents an approach to identify textual SCDs among the usual text of documents. It defines the problem of iSCD, where the words of SCDs are interleaved with the normal document text (content of document), and presents an approach to solve the problem. The approach uses the SCD-word distribution of a given corpus, which encodes the most likely words to occur with an SCD, as well as an HMM, which encodes being part of an SCD or not as the hidden state. Calculating a most probable sequence of hidden states in the HMM then allows for identifying the most probable windows for iSCDs. A case study on real-world and simulated data shows that the HMM-based approach has a robust performance in terms of recall and F1-score.

In future work, we aim to incorporate a person with a specific goal into the task of an agent that has a reference library with a set of SCDs. Additionally, we plan to put different SCDs into relation to each other through links, which, depending on the type of SCD could be text-based using similar words or relation-based using reoccurring entities or relations, for example. Links between SCDs further support the agent in providing information retrieval services.

Acknowledgment

The research is partially 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.

References

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

[2] F. Kuhr, B. Witten, and R. Möller, "On corpus-driven annotation enrichment," in *13th IEEE International Conference on Semantic Computing*. IEEE Computer Society, 2019.

[3] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[4] L. E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," in *Inequalities III: Proceedings of the Third Symposium on Inequalities*, O. Shisha, Ed. Academic Press, 1972, pp. 1–8.

[5] E. Wilden, *A Critical Edition and an Annotated Translation of the Akanāṉūṟu: Part 1, Kaliṟṟiyāṉainirai. Old commentary on Kaliṟṟiyāṉainirai KV - 90, word index of Akanāṉūṟu KV - 120*. École Française d'Extrême-Orient, 2018.

[6] S. Vijayarani, M. J. Ilamathi, and M. Nithya, "Preprocessing techniques for text mining-an overview," *International Journal of Computer Science & Communication Networks*, vol. 5, no. 1, pp. 7–16, 2015.

[7] Y. Sun, T. S. Butler, A. Shafarenko, R. Adams, M. Loomes, and N. Davey, "Word segmentation of handwritten text using supervised classification techniques," *Applied Soft Computing*, vol. 7, no. 1, pp. 71–88, 2007.

[8] M. A. Hearst, "Multi-paragraph segmentation of expository text," *arXiv preprint cmp-lg/9406037*, 1994.

[9] D. Dalva, U. Guz, and H. Gurkan, "Effective semi-supervised learning strategies for automatic sentence segmentation," *Pattern Recognition Letters*, vol. 105, pp. 76–86, 2018.

[10] I. Pak and P. L. Teh, "Text segmentation techniques: a critical review," in *Innovative Computing, Optimization and Its Applications*. Springer, 2018, pp. 167–181.

[11] F. Y. Y. Choi, "Advances in domain independent linear text segmentation," in *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, 2000. [Online]. Available: https://www.aclweb.org/anthology/A00-2004

[12] L. R. Rabiner and B. Juang, "A tutorial on hidden markov models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[13] J. Hu, M. K. Brown, and W. Turin, "Hmm based online handwriting recognition," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 18, no. 10, pp. 1039–1045, 1996.

[14] Y. Zhang, "Prediction of financial time series with hidden markov models," Ph.D. dissertation, Applied Sciences: School of Computing Science, 2004.

[15] N. Nguyen and D. Nguyen, "Hidden markov model for stock selection," *Risks*, vol. 3, no. 4, pp. 455–473, 2015.

[16] A. Srivastava, A. Kundu, S. Sural, and A. Majumdar, "Credit card fraud detection using hidden markov model," *IEEE Transactions on dependable and secure computing*, vol. 5, no. 1, pp. 37–48, 2008.

[17] M. Lange, F. Kuhr, and R. Möller, "Using a Deep Understanding of Network Activities for Workflow Mining," in *KI 2016: Advances in Artificial Intelligence - 39th Annual German Conference on AI, Klagenfurt, Austria, September 26-30*, ser. Lecture Notes in Computer Science, vol. 9904. Springer, 2016, pp. 177–184.

[18] T. Blum, N. Padoy, H. Feußner, and N. Navab, "Workflow mining for visualization and analysis of surgeries," *International journal of computer assisted radiology and surgery*, vol. 3, no. 5, pp. 379–386, 2008.

[19] R. Silva, J. Zhang, and J. G. Shanahan, "Probabilistic workflow mining," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 275–284.