Identifying and Translating Subjective Content Descriptions Among Texts

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

An agent pursuing a task may work with a corpus of documents as a reference library. Subjective content descriptions (SCDs) provide additional data that add value in the context of the agent's task. In the pursuit of 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 article presents a hidden Markov model-based approach to identifying SCDs in a new document where SCDs occur inline among content text. Additionally, we present a dictionary selection approach to identifying suitable translations for content text and SCDs based on n-grams. We end with a case study evaluating both approaches based on simulated and real-world data.

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, fulfilling a defined task, e.g., providing document retrieval services given requests from users. We assume that the corpus of an agent represents a specific context for the agent's task, since collecting documents is not an end in itself. Further, documents in a corpus might be associated with additional location-specific data making the content near 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 provides a value for different tasks of an agent, e.g., classifying new documents [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 and SCDs might be any form of additional data. 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 of doc-

uments. Consider *historical-critical editions* as an example, a document category where content and textual content descriptions can be interleaved, e.g., old poems in Tamil, which contain the poem itself and comments for specific words inline after the word added by editors or authors centuries later. The poem is the content and the comments are SCDs.

Only an agent with knowledge of the original poem can easily distinguish poem and comments. The first problem we tackle in this article is that of automatically identifying inline SCDs (iSCDs) among texts. 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 words in a document and words usually occurring with an SCD. If the words belong to the content, we expect that an 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 a word sequence belongs 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] to identify the iSCDs. Once iSCDs are identified, the agent can use them for further purposes, e.g., translating content and comments.

The second problem we tackle in this article is that of context-specific translation of content and SCDs, which may come from different contexts (or even centuries) and therefore may require different dictionaries. Assuming that an agent has translations for a set of documents, we can create dictionaries s.t. each dictionary contains for each word exactly one context-specific translation. For a new document with content and iSCDs identified, we present a context-specific dictionary selection approach based on n-grams, i.e., sequences of n neighboring words, to identify the best suited dictionary for the content and textual content descriptions.

Specifically, the contributions of this article are:

(1) an approach to identifying iSCDs among texts based on an HMM,
(2) an approach to identifying a suitable dictionary for content and iSCDs, and
(3) a case study based on real-world and simulated data to evaluate the performance of both approaches.

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 the n-gram-based approach to estimate suitable dictionaries to translate content and comments of a document. Afterwards, we evaluate the performance of both approaches. The case study is followed by a look at related work. Last, we conclude and present future work.

2. Preliminaries

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

2.1. *Subjective Content Descriptions*

We define the following terms to formalize the setting of a corpus.

- 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 documents of the same corpus represent a specific context. The term $\mathcal{V}_\mathcal{D}$ refers to the corpus-specific vocabulary of $\mathcal{D}$ representing the set of all words occurring in the documents of $\mathcal{D}$.
- For each corpus $\mathcal{D}$ exists a dictionary $dict(\mathcal{D})$ representing a suitable translation for each word in the documents of corpus $\mathcal{D}$. The set of all dictionaries is given by $Dict$.
- An SCD t represents a sequence of words $(w_1^d, \ldots, w_M^d)$, $M \in \mathbb{N}$, where each word $w_i^d \in t$ is from an SCD-specific vocabulary $\mathcal{V}_{g(d)}$, 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 $t \in g(d)$ exists an SCD window $win_{d,\rho}$ that refers to a sequence of words in d surrounding ρ in d. Mathematically, we can represent the window by $win_{d,\rho} = (w_{(\rho-i)}^d, \ldots, w_\rho^d, \ldots, w_{\rho+i}^d)$, $i \in \mathbb{N}$ 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 $pos(w^d, wind_{d,\rho})$ might be 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 $q = \frac{\rho}{n}$.

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

```
1:  function BUILDMATRIX(D)
2:      Input: Corpus D
3:      Output: SCD-word distribution 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)
```

2.2. *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 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:

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

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 tumbling 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 an approach to identifying iSCDs in documents.

3. Identifying Inline SCDs

In this section, we introduce the problem to identifying textual SCDs interleaved with content and present an HMM-based approach solving the problem.

3.1. *Inline SCD Problem*

The problem at hand consists of an agent being faced with a document containing content and iSCDs in the form of text 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 reconstruct the document's content text and use the located iSCDs for other purposes, e.g., to identify similar documents within the corpus. Given this setting, we introduce iSCDs into our notation as follows:

- Next to the corpus-specific vocabulary $\mathcal{V}_{\mathcal{D}}$, vocabulary $\mathcal{V}_{g(\mathcal{D})}$ represents the words occurring in (i)SCDs associated to documents in $\mathcal{D}$. Both vocabularies $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{g(\mathcal{D})}$ might overlap.
- An iSCD t is is represented by a sequence of words $(w_1, \ldots, w_n), n \in \mathbb{N}, w_i \in \mathcal{V}_{g(\mathcal{D})}$ and associated to the sequence of words preceding t in d.
- A document d containing iSCDs can be represented by sequences of words from $\mathcal{V}_{\mathcal{D}}$ and sequences of words from $\mathcal{V}_{g(\mathcal{D})}$ alternating, where the latter is associated with the preceding window of words.

Problem 1 introduces the inline SCD problem and Example 1 illustrates the inline SCD problem using a text containing two iSCDs.

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

Example 1 (Inline SCD Example). A document d contains the following sentence with two iSCDs, which are underlined:

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

However, the highlighting of iSCDs is not available in the original document s.t. an agent is faced only with the word sequence and has to decide which words

represent content words and which words represent iSCDs. It is easy the solve the problem if the two vocabularies do not overlap but becomes more difficult the more the two vocabularies share words.

We can reformulate Problem 1 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 belonging to content and another category represents the subsequences in d belonging to an iSCD.

3.2. *General Procedure*

To solve Problem 1, we work with two assumptions about corpus $\mathcal{D}$ and a new document d containing content text and iSCDs: (i) Document d belongs to the same context as $\mathcal{D}$. (ii) Vocabulary $\mathcal{V}_{\mathcal{D}}$ differs at least slightly from $\mathcal{V}_{g(\mathcal{D})}$. Given the first assumption, we can use $\mathcal{D}$'s SCD-word distribution $\delta(\mathcal{D})$ for d as well. Given the second assumption, $\delta(\mathcal{D})$ works well for estimating MPSCDs for content words and less well for iSCDs words. Estimating MPSCDs for d using a sliding window (instead of a tumbling one), we expect the following behavior: The MPSCD similarity value for a window over an iSCD should be significantly smaller than the MPSCD similarity value for a window over content. We can train an HMM with this behavior to solve Problem 1. Specifically, we need to perform the following steps:

(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:
    (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 S in the HMM given the sequence of similarity values of the MPSCD sequence as evidence and identify the words in d that belong to category "iSCD" given S.

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

3.3. *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 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 $w_{\sigma+1}^d$. We repeat this shifting operation until the end of the document, with the last window containing $(w_{(D-\sigma)}^d, ...w_D^d)$. For the sequence of words in each

Algorithm 2 Estimating MPSCD sequence

function ESTIMATEMPSCDSEQUENCE(d, σ, $\delta(\mathcal{D})$)
  Input: Document d, window size σ, SCD-word distribution $\delta(\mathcal{D})$
  Output: SCDs $g(d)$ with similarity values $\mathcal{W}$
  $\mathcal{W} \leftarrow \emptyset$
  for $\rho \leftarrow \frac{\sigma}{2}$; $\rho \leq \#word(d)$; $\rho \leftarrow \rho + 1$ **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 each word $w \in win_{d,t,\rho}$ **do**
      $\delta(win_{d,t,\rho})[w]$ += $I(w, win_{d,t,\rho})$
    $t \leftarrow \arg\max_{t_i \in g(\mathcal{D})} \frac{\delta(\mathcal{D})[t_i] \cdot \delta(win_{d,t,\rho})}{|\delta(\mathcal{D})[t_i]| \cdot |\delta(win_{d,t,\rho})|}$ in $win_{d,t,\rho}$
    $sim \leftarrow \max_{t_i \in g(\mathcal{D})} \frac{\delta(\mathcal{D})[t_i] \cdot \delta(win_{d,t,\rho})}{|\delta(\mathcal{D})[t_i]| \cdot |\delta(win_{d,t,\rho})|}$
    $\mathcal{W} \leftarrow \mathcal{W} \cup (sim, win_{d,t,\rho})$
    $g(d) \leftarrow g(d) \cup t$
  return $g(d)$, $\mathcal{W}$

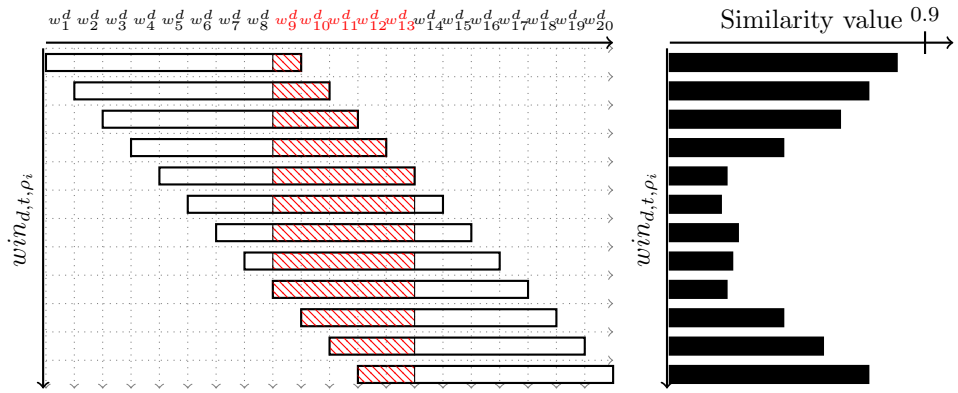


Fig. 1: Sliding window of size $\sigma = 9$ sliding over document d, where words w_9^d to w_{13}^d represent an iSCD, accompanied by corresponding similarity values on the right.

$win_{d,t,\rho}$, Alg. 2 calculates 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 representation of all SCDs in $\delta(\mathcal{D})$ using cosine similarity. Example 2 describes the sliding window behaviour and what similarity values might look like using an example setup.

Example 2 (Sliding Window and Similarity Values). 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 $win_{d,t_{12},w_{16}^d} = (w_{12}^d, ..., w_{20}^d)$. In this example, the words w_9 to w_{13} represent an

iSCD. Figure 1 also shows the sequence of MPSCD similarity values on the right for each window depicted to the left. We have a sequence of twelve MPSCD similarity values for the corresponding twelve windows. In the first window win_{d,t_1,w_5^d}, only 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 belong to an iSCD, the lower the corresponding window's MPSCD similarity value gets.

As shown in Fig. 1, iSCDs yield a specific pattern in the sequence of MPSCD similarity values: Similarity values first decrease, when the sliding window slowly moves into the iSCD, plateau, when the sliding window is squarely in the iSCD, and then increase again, with the the sliding window moving out of the iSCD. Next, we present an approach to solve Problem 1 by identifying iSCDs in d using an HMM trained of sequences of MPSCD similarity values.

3.4. *Estimating iSCDs*

We use an HMM to identify iSCDs using the sequence of MPSCD similarity values in d. First, we define the HMM and consider an example HMM for the iSCD problem. Second, we present how to identify iSCDs in d using an HMM.

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

- (hidden) states $\Omega = \{s_1, ..., s_n\}$; for the iSCD problem, $n = 2$, with state s_1 (s_2) meaning the words behind s_1 (s_2) belong to "content" ("iSCD"),
- an observation alphabet $\Delta = \{y_1, \ldots, y_m\}$, where each y_i represents a range of MPSCD similarity values; the observation alphabet is generated by discretizing MPSCD similarity values,
- a transition probability matrix A representing the probability of all possible state transitions $a_{i,j}, i, j \in \{1, 2\}$ between the two states $s_1, s_2 \in \Omega$, which implies moving forward in time from time step t to $t + 1$,
- an emission probability matrix B representing the probability of emitting a symbol from observation alphabet Δ for each possible state in Ω, and
- an initial state distribution vector $\pi = \pi_0$.

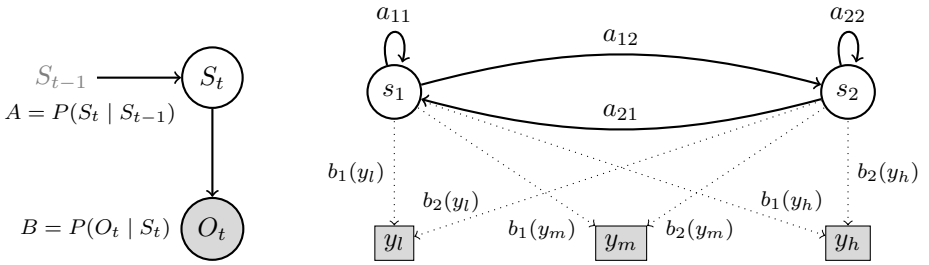
With $\sum_{j=1}^{n} a_{i,j} = 1$ for each $s_i \in \Omega$ summing over Ω, the entries of A between states $s_i, s_j \in \Omega$, represent the following conditional probability:

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

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

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

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



(a) One-time slice graph (b) State-transition system

Fig. 2: Hidden Markov model with one hidden variable S_t with two possible states $\{s_1, s_2\}$ emitting an observation O_t with three possible values $\{y_l, y_m, y_h\}$.

Example 3 (Graphical Representations). Figure 2 contains two graphical representations of an HMM $\lambda = (a_{ij}, b_j, \pi)$ with observation symbols $\{y_l, y_m, y_h\} \in \Delta$ and two states $\{s_1, s_2\} \in \Omega$. Figure 2a shows a one-time slice representation of the HMM. Figure 2b shows the HMM as the corresponding state-transition system.

In general, transition probability matrix A and emission probability matrix 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 MPSCD similarity value x to one of the m symbols in Δ. The specific discretization depends on the agent's task and can be adapted to each problem individually.

To solve Problem 1, we have to find the most likely sequence of states in an HMM λ, given a sequence of MPSCD similarity values $\mathcal{W}$. Algorithm 3 describes the workflow for identifying iSCDs based on such a sequence of similarity values $\mathcal{W}$ over the windows in d. First, Alg. 3 calculates the most likely state sequence by applying the discretization function f on the similarity values in $\mathcal{W}$, yielding an observation sequence O for λ. Then, it calculates the most likely sequence of states S in λ given O 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. Given S, Alg. 3 reconstructs the iSCDs by identifying the windows behind those states in S that are equal to s_2 ("iSCD") and marking the words in each corresponding window as an iSCD. Example 4 illustrates estimating the most likely sequence of states using the Viterbi algorithm for MPSCD similarity values resulting from the twelve windows shown in Example 2.

Example 4. Let us assume that Alg. 2 yields the following MPSCD similarity values for the 12 windows in Example 2, as shown in Fig. 1:

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

Algorithm 3 Estimate iSCDs using MPSCD similarity values and a trained HMM

function ESTIMATEISCDS(λ, f, $\mathcal{W}$, d, σ)
  Input: HMM λ, discretization function f, similarity values $\mathcal{W}$, document d, window size σ
  Output: SCDs $g(d)$
  $O \leftarrow ()$
  for each similarity value $sim \in \mathcal{W}$ **do**
    $O \leftarrow O \circ f(sim)$
  $S \leftarrow$ VITERBI(λ, O) ▷ Compute the most likely state sequence
  $g(d) \leftarrow \emptyset$
  for each $i, state \in S$ **do** ▷ Iterate over S, maintain an index i
    if $state = s_2$ **then** ▷ s_2 corresponds to "iSCD"
      $t \leftarrow (w_i^d, \ldots, w_{i+\sigma}^d)$
      $g(d) \leftarrow g(d) \cup t$
  return $g(d)$

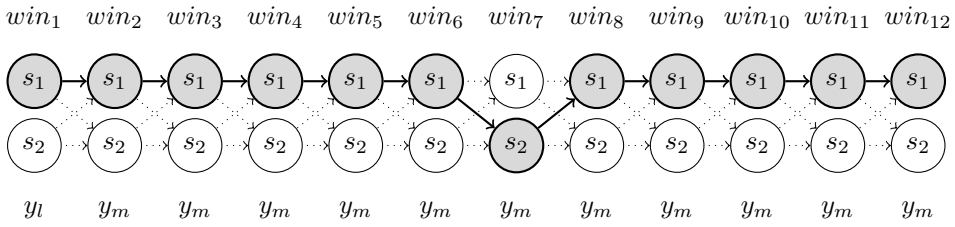


Fig. 3: Trellis corresponding to the MPSCD sequence of Example 2.

Using the following function for discretization

$$f(x) = \begin{cases} y_l & 0 \le x < \theta_1 \\ y_m & \theta_1 \le x < \theta_2 \\ y_h & \theta_2 \le x \le 1, \end{cases} \tag{2}$$

where $\theta_1 = 0.3$ and $\theta_2 = 0.7$, we get the following 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 3 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 iSCD 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 $(w_9^d, \ldots w_{13}^d)$, which make up the iSCD in the example.

Correctness By identifying the word sequences that are most probably iSCDs, we automatically identify which words belong to iSCDs and which words belong to content. 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 an approach to selecting dictionaries for content text and iSCDs, which may have been identified using the method just presented.

4. Context-specific Dictionary Selection

In the last section, we have presented an HMM-based approach to identifying iSCDs among texts. In this section, we introduce the problem of selecting a suitable dictionary to translate both, iSCDs and content. Additionally, we present an approach to solve the problem by automatically selecting suitable dictionaries.

4.1. *Dictionary Selection Problem*

An agent interested in translating a document has to decide on a dictionary for unknown words to select a suitable translation. Generally, there is no single translation for a word since the translation of a word depends on the specific context. Consider historical-critical editions representing poems as content and iSCDs as comments. Translating historical-critical editions is non-trivial since the edition contains comments written from different authors at different times. Using a single dictionary to translate the text of a historical-critical edition might result in non-optimal translations because a single dictionary ignores the fact that a poem has been enriched with additional descriptions at different points in time. Thus, we are interested in selecting for a poem and each identified iSCD the corresponding Q best suited dictionaries. Generally, the dictionary selection problem asks for the top-Q dictionaries for a sequence of words $(w_1, \ldots, w_i)$ representing the poem or an iSCD, given J corpora $\{\mathcal{D}\}_{i=1}^{J}$ containing a set of documents and their corresponding translations. Equation (3) represents a mathematical description of the document selection problem.

$$\underset{dict_1,\ldots,dict_Q \in Dict}{\arg\max} P(dict_1, \ldots, dict_Q | (w_1, \ldots, w_i), \{\mathcal{D}\}_{i=1}^{J}) \tag{3}$$

4.2. *Dictionary Selection Approach*

The dictionary selection problem, introduced in Section 4.1, asks for the Q most suitable dictionaries for a sequence of words in a new document d given a set of corpora containing documents and their corresponding translations. The sequence of

Algorithm 4 N-gram-based Dictionary Selection

function SELECTDICTIONARY(n, $\{\mathcal{D}\}_i^J$, seq)
  Input: length of n-gram n, set of corpora $\{\mathcal{D}\}_i^J$, word sequence seq
  Output: suitable dictionary: $dict$
  $v_{seq} \leftarrow$ n-gram-frequency of seq
  $s \leftarrow 0$
  $dict \leftarrow$ empty
  for each corpus $\mathcal{D} \in \{\mathcal{D}\}_i^J$ **do**
    $v_{\mathcal{D}} \leftarrow$ corpus-specific n-gram-frequency
    $sim \leftarrow \frac{v_{\mathcal{D}} \cdot v_{seq}}{|v_{\mathcal{D}}| \cdot |v_{seq}|}$
    if $sim > s$ **then**
      $dict \leftarrow dict(\mathcal{D})$
      $s \leftarrow sim$
  return $dict$

words might represent the poem or an iSCD. Generally, different translations might exist for a word and a suitable translation is only possible if the corresponding context is available. For a new document d containing only words, we can represent the context of each word by its neighbouring words. In this section, we present an approach to identifying the Q most suitable dictionaries to translate a word sequence available in d based on the n-grams occurring within the documents of the J corpora and the n-grams in an iSCD. Thus, we create for each word in a sequence of words the n-grams by using the neighbouring words and identify a suitable dictionary for a sequence of words based on the similarity of all sequence-specific n-grams and all n-grams we can generate from documents of all J corpora. Finally, we select the dictionary associated to the corpus containing the most similar n-grams. Let us assume we have a set of J corpora and for each corpus $\mathcal{D}$ exists a corpus-specific dictionary that has been used to translate the documents, e.g., dictionary $dict_1$ is suitable for documents in $\mathcal{D}_1$ and another dictionary $dict_2$ suitable for documents in corpus $\mathcal{D}_2$. Additionally, we assume a new document d is available and using Alg. 3 we have successfully identified the content and comments. Then, we are interested in selecting a suitable dictionary for the original text in d and a suitable dictionary for each iSCD based on the n-grams we can create from the sequences of words referring to the text and to the iSCDs.

Algorithm 4 represents the technical description for selecting a suitable dictionary for a word sequence seq. The inputs are the length n of n-grams, all corpora $\{\mathcal{D}\}_i^J$, and a a word sequence seq. The output of Alg. 4 is the best suited dictionary $dict$. Algorithm 4 starts by calculating the frequency v_{seq} of n-grams available in word sequence seq. Then, Alg. 4 calculates the n-gram frequencies $v_{\mathcal{D}}$ for each corpus $\mathcal{D}$ by analyzing each document in $\mathcal{D}$ and computes the cosine similarity between v_{seq} and $v_{\mathcal{D}}$. Alg. 4 stores the current corpus' dictionary if it has a higher similarity value as any corpus checked before. In the end, Alg. 4 returns the stored dictionary.

Correctness We can calculate all possible n-grams from a given sequence of words representing an iSCD or a poem. Additionally, we calculate all possible n-grams for each document in the J corpora. Thus, we can compare the n-gram frequency vectors from a sequence of words of new document d with each corresponding n-gram frequency vector of each corpus. Using a vector representation of the n-gram frequency allows us to compare the frequencies based on the cosine similarity used in Line 9 of Alg. 4. Comparing the n-gram frequency vector of a word sequence with the n-gram frequency vectors of each corpus yields the most similar corpus. Since each corpus is linked to a corpus-specific dictionary, we can directly select the best suited dictionary for each sequence of words in d.

5. Case Study

After introducing the HMM-based approach to identifying iSCDs among texts, followed by the n-gram-based dictionary selection approach, we present a case study illustrating the potential of both approaches estimating iSCDs within documents of different data sets as well selecting suitable dictionaries. We start with a description of data sets we use in this case study followed by the workflows for analyzing the performance of the HMM-based iSCDs detection approach and the n-gram-based dictionary selection. Additionally, we illustrate the potential of both approaches by evaluating the performance on the data sets.

5.1. *iSCD Detection*

In this section, we present the data sets, the workflow and results for the iSCD detection approach introduced in Section 3.

5.1.1. *Data Sets*

We use the following seven data sets to evaluate the performance of the HMM-based approach to identifying iSCDs in a new document.

(1) `Tamil` consists of 91 poems transcribed from old palm leaves [5].
(2) `Greek` consist of 1 treatise about Aristotle's Categories [6].
(3) `US` consists of 74 articles about cities in the Unites States of America.[a]
(4) `EU` consists of 10 articles about cities in Europe.[b]
(5) `Arxiv-general` consists of 500 randomly selected abstracts from Arxiv[c].
(6) `Arxiv-CS` consists of 500 randomly selected abstracts from publications of the Computer Science (CS) category available on Arxiv[c].
(7) `Newsgroups` consists of 290 posts from the 20 newsgroups data set containing 18.828 newsgroup posts on 20 topics [7].

[a]US cities – `https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population`
[b]European cities – `https://en.wikipedia.org/wiki/List_of_urban_areas_in_Europe`
[c]ArXiv – `https://www.kaggle.com/Cornell-University/arxiv`

Table 1: Characteristics of data sets divided into eight different settings.

	Tamil		Greek	US	EU	Arxiv		News-
	Unst.	St.				general	CS	groups
$\lvert\mathcal{D}\rvert$	91	91	1	74	10	500	500	290
Avg $\#word(d)$	73,2	73,2	10.458	200,7	318,7	74,4	77,3	133,7
# iSCDs	908	869	143	1.814	757	4.905	4.715	6.541
$\lvert\mathcal{V}_\mathcal{D} \cup \mathcal{V}_{g(\mathcal{D})}\rvert$	8.015	5.783	3.623	6.688	3.314	10.083	8.843	12.829
$\lvert\mathcal{V}_\mathcal{D}\rvert$	5.521	4.304	1.894	3.657	1.439	5.776	4.715	7.558
$\lvert\mathcal{V}_{g(\mathcal{D})}\rvert$	2.666	1.987	2.123	3.902	2.232	6.751	6.493	8.332
$\lvert\mathcal{V}_\mathcal{D} \cap \mathcal{V}_{g(\mathcal{D})}\rvert$	172	508	394	871	357	2.444	2.365	3.061

For the `Tamil` and `Greek` data sets, we extract the documents from [5] and [8], respectively. [8] contains a digital copy of the Aristotle's Categories we have used to extract the text. Each document is associated with comments about the content from the original documents and acting as iSCDs. The `US` and `EU` data sets consist of articles from the open and widely accessible online encyclopedia *Wikipedia* containing text about cities in the US and EU, respectively. The `Arxiv` data set contains a subset of 500 randomly selected abstracts from the Arxiv library. Additionally, for the `Arxiv-CS` data set we randomly selected 500 abstracts from the Arxiv library referring to the CS category. The `Newsgroups` data set represents a randomly selected subset of the well-known 20 newsgroups data set.

In contrast to the documents in `Tamil` and `Greek`, the documents in the other data sets do not contain iSCDs. Thus, we generate iSCDs for each document of data sets (3) - (7) by (i) downloading a dump of the English free online dictionary *Wiktionary*, (ii) creating an SCD for each word in a document using the corresponding Wiktionary entry (if available), and (iii) splicing the SCDs into the document.

After we have downloaded all documents and determined iSCDs we store the documents in the respective corpora. Next, we use the following standard preprocessing tasks from the NLP community, (i) lowercasing all characters, (ii) stemming the words, (iii) tokenizing the result, and (iv) eliminating tokens from a stop word list containing 179 words, to transform the text of the documents into a more digestible form for machine learning algorithms to increase their performance [9]. Generally, language changes over time, and modern word stemmers and stop words might not be optimal for old texts. Thus, we neither eliminate stop words nor stem words in `Greek`. For `Tamil`, we compare the performance of the unstemmed (unst.) corpus to a stemmed (st.) corpus by a modern word stemmer[d] for Tamil.

Table 1 gives an overview about (i) the number of documents in a corpus, (ii) the average length of documents, and (iii) the size of different vocabularies. Stemming

[d]`https://github.com/rdamodharan/tamil-stemmer`

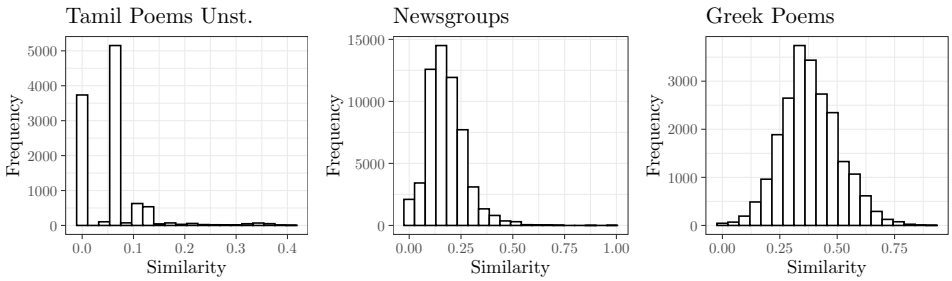


Fig. 4: Histograms of similarity values gained from Alg. 2 for unstemmed `Tamil`, `Newsgroups`, and `Greek`. *Note the different scaling of the axes.*

the poems in `Tamil` yields to less different words in the vocabulary and more intersecting words between content (poem) and their comments (iSCDs). For ease of reading, we divide numbers with many digits into group using a dot as delimiter.

5.1.2. *iSCD Detection Workflow*

Evaluating the performance of the HMM-based approach to identifying SCDs among texts requires a setup of the HMM. We use two hidden states (s_1,s_2) as defined in Def. 1 and five observable states y_1 to y_5. We have tested different setups and five observable symbols having yielded good results. The thresholds are selected by analyzing histograms of similarity values. Figure 4 represents histograms of similarity values gained from Alg. 2 for different corpora. The intervals for y_1 to y_5 should be selected from the areas of similarity values having a high frequency to gain a good HMM-based iSCD detection performance. For all data sets expect `Greek`, we select the intervals: $y_1 = [0.0, 0.05)$, $y_2 = [0.05, 0.1)$, $y_3 = [0.1, 0.15)$, $y_4 = [0.15, 0.20)$, and $y_5 = [0.20, 1.0]$. For `Greek`, we use the intervals: $y_1 = [0.0, 0.3)$, $y_2 = [0.3, 0.35)$, $y_3 = [0.35, 0.4)$, $y_4 = [0.4, 0.45)$, and $y_5 = [0.45, 1.0]$.

We perform the following five tasks on each data set to evaluate the performance of the HMM-based iSCD detection approach:

(1) Split the data set into multiple parts and use leave-one-out cross-validation resulting in training sets of 80-90% of the documents and test sets containing the remaining 10-20% of the documents.
(2) Form the SCD-word distribution δ for the training set using Alg. 1.
(3) Generate an HMM and apply the Baum-Welch algorithm [3] to train the hidden parameters of the HMM.
(4) For each document in the test set estimate MPSCDs using Alg. 2, and
(5) Compute the most probable sequence of hidden states in the HMM for the discretized sequence of similarity values using the Viterbi algorithm.

We use the F_1-score to evaluate the performance, which is defined by:

$$F_1\text{-Score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}},$$

where precision and recall are defined by: $\frac{tp}{tp+fp}$ and $\frac{tp}{tp+fn}$, respectively, with $tp = \#$ true positives, $fp = \#$ false positives, $tn = \#$ true negatives, and $fn = \#$ false negatives. We use cross-validation and the total average of the precision, recall and F_1-score in the results.

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. We use the histograms in Fig. 4 to choose the values of ℓ. For unstemmed `Tamil`, $\ell = 0.05$ and for `Greek`, $\ell = 0.35$ results in best performance for iSCD detection. For all other data sets, $\ell = 0.1$ yields the best results.

As depicted in Table 1, unstemmed `Tamil` contains only 172 words, representing 2% of all words, that occur in both vocabularies $\mathcal{V}_\mathcal{D}$ and $\mathcal{V}_{g(\mathcal{D})}$. After stemming the words in `Tamil`, this share increases to 8%. In `US`, `EU` and `Greek` $\mathcal{V}_\mathcal{D}$ and $\mathcal{V}_{g(\mathcal{D})}$ share around 10% of their words, while `Arxiv` and `Newsgroups` share around 25% of their words, respectively. Thus, we expect a good word-based classification performance for data sets sharing less words between $\mathcal{V}_\mathcal{D}$ and $\mathcal{V}_{g(\mathcal{D})}$, e.g., for `Tamil`.

Next, we present the results for the HMM-based iSCD detection approach.

5.1.3. *Results*

Figure 5 presents the performance of the HMM-based, word-based, and threshold-based approach for all data sets. For the HMM-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:

$$\{y_1 : 0.15, y_2 : 0.50, y_3 : 0.20, y_4 : 0.10, y_5 : 0.05\}$$

and in case of text:

$$\{y_1 : 0.05, y_2 : 0.05, y_3 : 0.30, y_4 : 0.35, y_5 : 0.25\}.$$

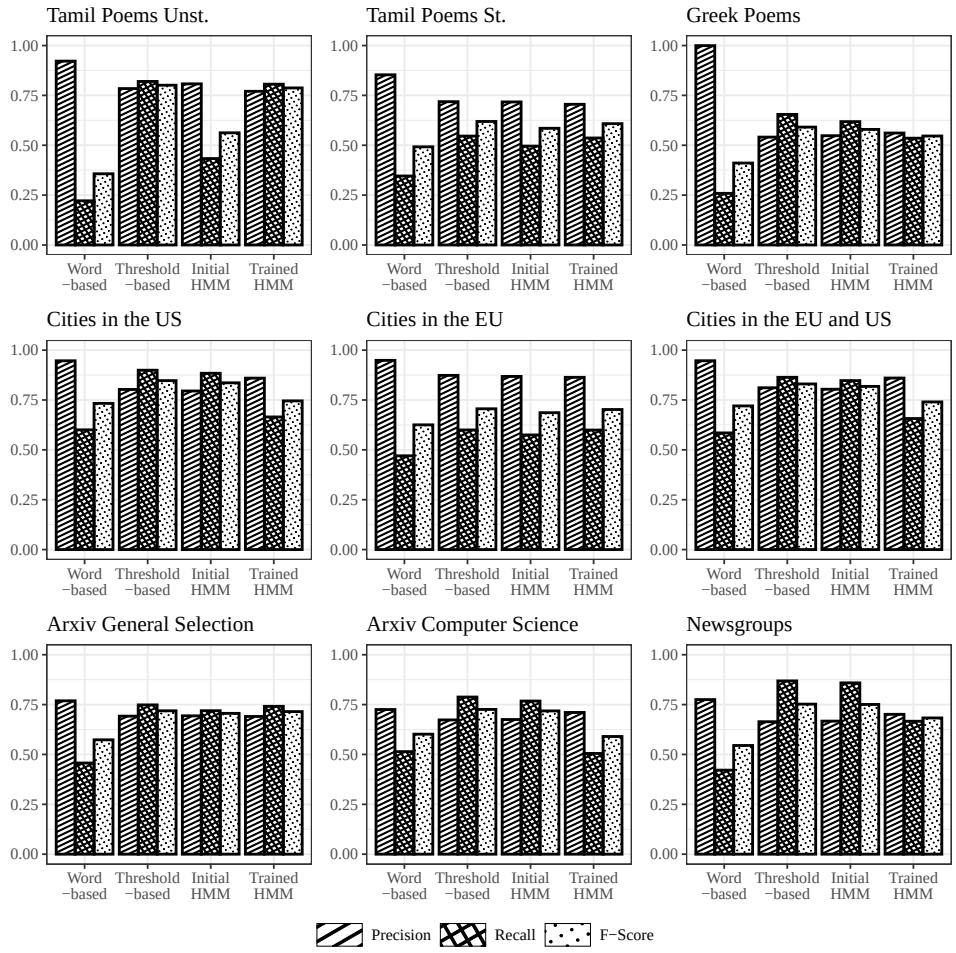


Fig. 5: Classification performance for HMM-, word-, and threshold-based approach.

Unstemmed `Tamil` yields very low MPSCD similarity values (Fig. 4) in contrast to the other data sets yielding values similar to `Newsgroups`. Therefore, we slightly modify the emission probabilities for unstemmed `Tamil` by increasing y_1 and y_2. The emission probabilities encode that a window associated with an MPSCD of a high similarity value is unlikely being classified as an iSCD.

The word-based classification yields the best F_1-Score for `US`. The precision of the word-based classification is very high for `Tamil` and `Greek`, while the precision is low for `Arxiv`. This corresponds to our assumption made before: Fewer shared words between $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{g(\mathcal{D})}$ result in better precision. The recall of the word-based classification is considerably lower than the recall of the HMM-based approach. Interestingly, the word-based classification performance is not as poor as expected

for data sets containing more overlapping vocabulary between $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{g(\mathcal{D})}$.

The threshold-based classification performance is good for all data sets. Often the results are similar or even better than the results of the HMM-based approach. However, for the threshold-based approach it is difficult to determine the best values ℓ for the threshold, since the threshold changes for each data set.

The HMM-based approach performs well for all data sets. Mostly, the trained HMM, the initial HMM, and the threshold-based classification result in similar values. The difference between the initial and the trained HMM is not as pronounced since the initial values for the emission probabilities are already of good quality, which reduces not only the runtime for learning but also has the effect of the initial model performing relatively well. The discretization function uses intervals close to threshold ℓ, such that threshold-based classification performs similar well, too.

Overall, the HMM-based approach yields the best performance in the case study. Even thought, the threshold-based classification sometimes results in slightly better results for most data sets. However, choosing a threshold is difficult and training an HMM is much easier.

5.2. *Dictionary Selection*

This section presents the data sets, the workflow and results for the dictionary selection approach.

5.2.1. *Data Sets*

For the evaluation of the dictionary selection approaches, we use the entire 20 Newsgroups data set mentioned above, containing 18.828 documents. On average each document contains 16 sentences and each newsgroup features 4.942 unique words while sharing 8.639 words with other newsgroups.

Additionally, we use a data set (`Shakespeare`) containing 39 Shakespeare documents taken from *Project Gutenberg*[e]. On average, each document in `Shakespeare` consist of 2.526 sentences. The total number of different words in the vocabulary of both data sets is 139.788, while both data sets share only 9.183 words. 9.826 words occur only in `Shakespeare`, and 120.779 words occur only in documents of the 20 newsgroups data set.

For documents in both data sets, we apply the same four preprocessing tasks from the NLP community already used for the *iSCD Detection*, namely lowercasing all characters, stemming the words, tokenizing the result, and eliminating tokens from a stop word list.

[e]https://www.gutenberg.org/

5.2.2. *Dictionary Selection Workflow*

Given multiple dictionaries and for each dictionary a well suited set of documents, the dictionary selection problem asks for the best suited dictionaries to translate a document d. In our scenario d is a poem interleaved with iSCDs. After extracting the content (d_p) and the set of iSCDs ($g(d)$) using the HMM-based iSCD detection approach, we perform the following steps to translate the content in d.

(i) Specify the length of n-grams we use in Alg. 4.
(ii) Perform Alg. 4 to identifying a suitable dictionary for content and iSCDs.

We run all experiments on a virtual machine featuring 8 Intel 6248 cores at 2.50GHz (up to 3.90GHz) and 16GB RAM.

Using the two data sets we demonstrate the performance of the n-gram-based document selection approach in two different settings: First, we are interested in selecting the best dictionary among a modern language dictionary for all 20 newsgroups and an ancient language dictionary for `Shakespeare`. Second, we assume to have a context specific dictionary for each single newsgroup and we are interested in selecting the best dictionary among the 20 dictionaries.

For each setting we split the data sets randomly into a training set containing 80% of the sentences and a test set containing the remaining 20%. After training the models we measure the accuracy as proportion of correctly selected dictionaries for the sentences in the test set.

We compare the performance of the n-gram-based approach, introduced in Section 4.2, with the well-known Skip-gram model [10, 11]. The training objective of the Skip-gram model is to find word representations that are useful for predicting the surrounding words in a sentence or a document. Given a sequence of training words, the objective of the Skip-gram model is to maximize the average log probability. We generate for each corpus a Skip-gram model and use distance measures to calculate the distance between a corpus-specific model and a given sentence from d to identify a suitable dictionary for a sentence based on the corpus where the corresponding model has the shortest distance. We analyse the performance of the following distance measures:

Jaccard Given a sentence, we take each word and query the model for the surrounding words. The words predicted are then compared to the actual words in the sentence using the Jaccard index. We use the mean of all Jaccard indices to get the distance between the sentence and a model.

Model Given a sentence of a document we calculate the distance of each word of the sentence to all words known by the model. To get the distance between the sentence and the model we use the mean of the distances for each word.

Sentence This approach is similar to *Model*, but we do not calculate the distance to all words known by the model, but only to the words occurring in the sentence. Thus, we also take each word of the given sentence and calculate the distance within the model to the other words in the given sentence.

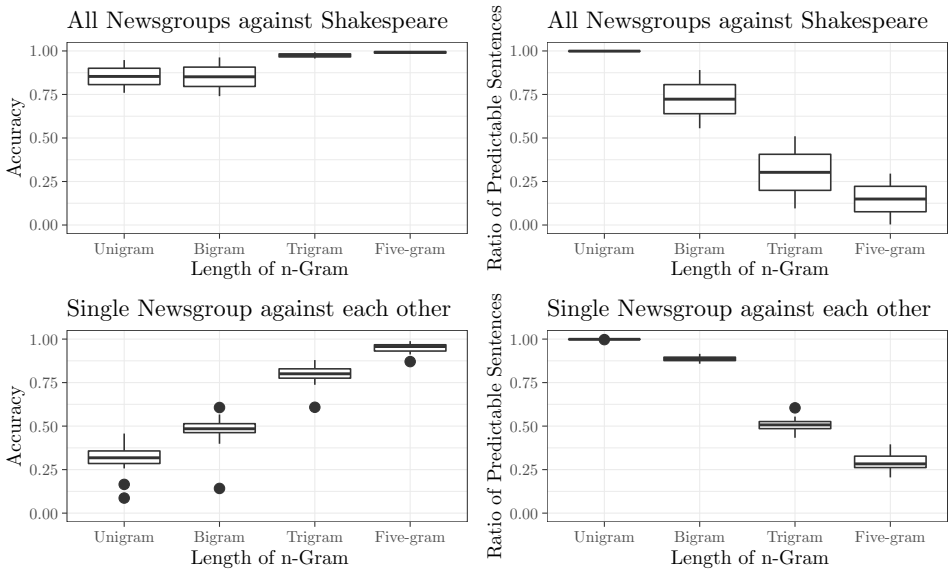


Fig. 6: Accuracy (left) and ratio of predictable sentences (right) of the n-gram-based approach for different data sets and different lengths of n-grams.

Voting Calculate all three distance measures (Jaccard, Model, Sentence), predict the most probable model for each of the three measures and perform a majority voting. To determine the top-n best matching dictionaries, we predict the n most probable models for each distance measure and then perform a majority voting across the $3n$ predictions.

5.2.3. *Results*

Figure 6 shows boxplots of accuracies and ratio of predictable sentences. The ratio indicates for how many sentences the model could explicitly select a dictionary for, i.e., when using five-grams it may happen that a sentence contains no five-gram known by the model and therefore no selection is possible.

In both settings, the model is able to predict all sentences using unigrams, but not while using five-grams. For `Shakespeare`, the accuracy is overall high, confirming that modern and ancient languages feature more differences than different contexts. For `Newsgroups`, the accuracy grows with the length of the n-grams while the ratio decreases. Generally, bigrams provide a good compromise between accuracy and ratio. While rating the reached accuracy of 0.5 when using bigrams, we have to remember that randomly choosing a dictionary among 20 would result in an accuracy of around 0.05%.

As stated before, we compare the performance of the n-gram-based approach to the Skip-gram-based approach. In Fig. 7, we show the accuracy and ratio of the

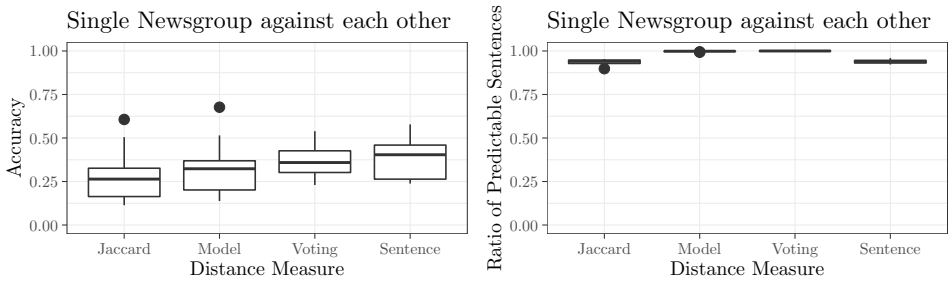


Fig. 7: Accuracy (left) and ratio of predictable sentences (right) of the Skip-gram-based approach on different newsgroups.

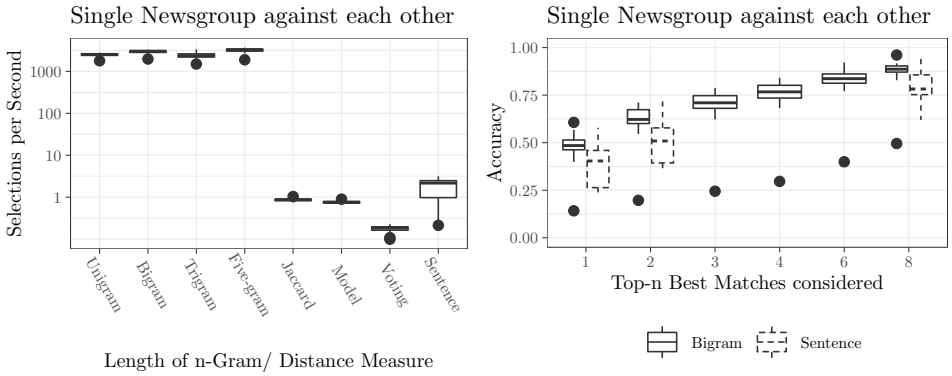


Fig. 8: Speed (left) of both approaches shown as number of sentences processed per second (logarithmic scaled). Top-n performance (right) using the best parameters of each approach.

Skip-gram-based approach. The Skip-gram-based approach generally yields lower accuracy but provides higher ratios than the n-gram-based approach. For the Skip-gram-based approach, the `Sentence` distance measure results in the best accuracy. Comparing values for the best length of the n-grams with the best distance measure, bigrams and `Sentence` respectively, the accuracy of the n-gram-based approach is clearly higher while both approaches result in similar ratios.

In Fig. 8, we illustrate the runtime of both approaches. On the left side the performance of both approaches in term of selections performed per second. The n-gram-based approach performs around 3.000 selections per second while the Skip-gram-based approach only performs less than 10 selections in the same time.

Last but not least, we present the performance of the top-n best matching

dictionaries on the right side of Fig. 8 since an agent might be allowed to return a ranked list of results for a task. We only consider bigrams and the `Sentence` distance measure. An accuracy clearly above 0.5 is already achieved by only considering the top-2 dictionaries. Overall, the n-gram-based approach shows a good performance, especially when using bigrams. In comparison to the Skip-gram-based approach, short run times and good accuracy results are the advantages.

6. Related Work

We look at related work in the area of text segmentation, HMM-based classification, as well as word embeddings.

Text Segmentation In text segmentation, the task is to separate text into segments, such as words [12], topics [13], sentences [14], passages, or lines. The difficulty lies in identifying the segment borders [15]. In case of topic segmentation, each coherent sequence of words that is part of the content or part of an iSCD, is a segment. In topic segmentation, Hearst [13] uses a sliding window over a sequence of words and compares adjacent windows with the current window. The more the adjacent windows are similar, the more it is likely that a subtopic continues. Transferred to our setting, the inference would be that text (or iSCD) continues on as the window slides on and the similarity remains high. Choi [16] 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.

Hidden Markov Model 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 [17], character recognition [18], finance data prediction [19, 20], credit card fraud detection [21], and workflow mining [22, 23, 24]. With the identification of iSCDs, we have successfully applied HMMs in another context, solving a new task.

Word Embedding Word embedding is a class of representation of words for text analysis. Often, words are represented as real-valued vector that encodes the meaning of the word. Words that are close to each other in the vector space are expected

to be similar in meaning. Since the 1990s, vector space models have been used in distributional semantics and different models have been introduced, e.g,. LSI [25] or LDA [26]. Bengio et al. have coined the term *word embedding* [27]. They have trained a word embedding in a neural language model together with model parameters. Collobert and Weston demonstrate the power of pre-trained word embeddings in [28].

In 2013, Mikolov have introduced word2vec [11] – enabling training and use of pre-trained embeddings. More recent word embedding approaches providing pre-trained models for different NLP tasks include the transformer-based language models BERT [29] and GPT [30].

7. Conclusion

This paper presents an approach to identifying textual SCDs among the usual text of documents as well as an approach to identifying suitable dictionaries to translate content and SCDs. We define the iSCD problem, where the words of SCDs are interleaved with the normal document text (content of document), and present 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.

Additionally, we have introduced the context-specific dictionary selection problem and present an n-gram-based dictionary selection approach to identify the most suitable dictionary for each iSCD and the content of a document. A case study on real-world and simulated data shows the performance for both approaches 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. 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. The authors thank José Maksimczuk for providing the `Greek` data set as well as Eva Wilden and Giovanni Ciotti for providing the `Tamil` data set.

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] I. Bekker *et al.*, "Aristotelis opera edidit academia regia borussica," *Volumen primum-Volumen alterum*, 1831.

[7] "20 newsgroups data set." [Online]. Available: http://qwone.com/~jason/20Newsgroups/

[8] Alignment of aristotle's categories. Accessed: 2021-04-09. [Online]. Available: http://textalign.net/output/ar.cat.tan-a-div-collated-obj.html

[9] 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.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[11] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *arXiv preprint arXiv:1310.4546*, 2013.

[12] 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.

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

[14] 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.

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

[16] 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

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

[18] 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.

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

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[26] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[27] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *The journal of machine learning research*, vol. 3, pp. 1137–1155, 2003.

[28] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167.

[29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[30] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.