International Journal of Semantic Computing
© World Scientific Publishing Company

Augmenting and Automating Corpus Enrichment

Felix Kuhr, Tanya Braun, Magnus Bender, Ralf Möller
Universität zu Lübeck
Institute of Information Systems
Ratzeburger Allee 160, 23562 Lübeck, Germany
{*kuhr,braun,m.bender,moeller*}*@ifis.uni-luebeck.de*
http://www.ifis.uni-luebeck.de

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

An agent in pursuit of a task may work with a reference library containing documents associated with additional data that provide location-specific explanations about the content. Faced with a new document, an agent has to decide whether to include the new document in its reference library. Basing the decision on words, topics, or entities has shown not to lead to a balanced performance for varying documents. In this article, we present an approach for automatically enriching new documents with data associated to documents in a reference library. Additionally, we analyse these data to classify new documents into categories to help an agent in deciding whether to include the new document in its reference library.

Keywords: Subjective Content Descriptions; Corpus Enrichment; Text Mining.

1. Introduction

An agent in pursuit of a task, explicitly or implicitly defined, may work with an individual set of documents (corpus) as a reference library. We assume that the individual collection of documents represents a specific context in which an agent performs its task. From an agent-theoretic perspective, an agent is a rational, autonomous unit acting in a world, perceiving through sensors, and fulfilling a defined task, e.g., providing document retrieval services given requests from users. We assume that documents in a corpus are associated with additional location-specific data making the content explicit by providing descriptions, references, or explanations about the content in documents, and we denote these data as subjective content descriptions (SCDs).

The documents in a corpus play an important role for an agent's task, and the associated SCDs are optimized for the task. An agent might be interested in extending a corpus with a new document only if the content of the new document is relevant for the task of the agent, e.g., the new document contains content related to the content of other documents in the corpus. But, what should an agent do if presented with a new document, which typically has no associated SCDs? The question for the agent is: Does the content of a new document provide anything of value to add in the context of a given corpus? Generally, an agent can classify new documents into categories, where each category represents the relationship between the content of new documents and the content of documents in the corpus. We use the following four document categories: (i) *sim* - new document is similar to at least one document in the corpus, (ii) *ext* - new document is an extension of a document in the corpus, (iii) *rev* - new document is a revision of a document in the corpus, and (iv) *unrel* - new document is unrelated to all documents in the corpus. Additionally, we can argue that a new document provides a value for the task of an agent if the document is classified into category *sim*, *ext*, or *rev*, since a document classified to one of those three categories contains content related to content of documents in the corpus. We denote the process of extending a corpus with documents adding value with the notion *corpus enrichment*.

Given a corpus containing documents associated with SCDs, we can correlate SCDs and words in a window around the SCDs from documents in the corpus resulting in SCD word frequency vectors, one vector for each SCD. We use a word frequency vector to represent each SCD instead of a bit vector, since SCDs are not unique and a SCD might occur more than once. Faced with a new document having no SCDs, an agent can divide a new document into windows and generate a word frequency vector from the words in each window to estimate the most probably suited SCDs (MPSCDs) for the document by comparing the word frequency vector of each window with the word frequency vector of each SCD of the corpus.

In this work, we use the cosine similarity to compare the word frequency vector of a window with the word frequency vectors of all SCDs associated to documents in the corpus by calculating the distance between the vectors. We add the SCD to a window where the corresponding word frequency vector has the smallest distance (highest cosine similarity) to the word frequency vector of the window and we denote the SCD as the MPSCD. We argue to divide a new document into windows and estimate for each window the corresponding MPSCD resulting in a sequence of MPSCDs, where each MPSCD is associated with a location in the new document and contains a similarity value. For a new, *unrelated* document even the vectors associated to the MPSCDs have a large distance to the word frequency vector of the new document resulting in a sequence of low similarity value.

We assume that the relatedness of a window's content to the task of an agent has an impact on the relatedness of the content in the next window. Different approaches exist to model the sequential impact. In this article, we use hidden Markov models (HMMs) modelling the sequential impact and use the relatedness of the content (related/unrelated) as hidden states and MPSCD similarity values as observations.

To estimate the category of a new document, we perform the following steps. First, we learn for each of the four document categories a corpus-specific HMM where each slice in an HMM concerns a window in a document and the hidden state in the HMMs concerns whether the SCD in the window is related or unrelated to the agent's task. Second, given a new document, we estimate for each window an MPSCD and use the corresponding sequence of MPSCD similarity values as observable input data for the Viterbi algorithm [1] to estimate the most likely sequence of related/unrelated content (hidden states) for each of the four previously learned HMMs. This technique results in four probability values, one for each of the estimated most likely sequences of related/unrelated content. We can use the probability values to rank the most likely sequences. The category of the new document is given by the category corresponding to the HMM resulting in the sequence with highest probability.

Additionally, an agent can use the most likely sequence of hidden states to augment corpus enrichment with providing location-specific information about new data within the context of the specific corpus. If the agent decides to extend its corpus with new documents based on the documents' category it can even choose to retain the initially estimated MPSCDs, possibly adapting them with new SCDs, and use them as a basis for enriching SCDs in the corpus, in an automatic way [2,3].

Specifically, the contributions of this article are:

(i) Solving the problem of classifying new documents in a context-specific way.
(ii) Providing a decision making procedure for corpus enrichment based on the classification of new documents.
(iii) Presenting two approaches for optimizing the quality of initially estimated SCDs for new documents by (a) adapting the size and position of MPSCD windows in a new document, and (b) considering only a subset of documents from the corpus having a high similarity with new documents, and
(iv) presenting a case study regarding the classification performance of new documents for three different corpora as well as the performance of the two optimization techniques.

The remainder of this article is structured as follows: We start with a look at related work. Then, we specify notations and recap the purpose of SCDs. Next, we present HMM-based document classification and the decision making procedure for corpus enrichment. We also discuss augmenting corpus enrichment by providing location-specific information about new data, followed by optimization techniques for SCDs. Afterwards, we present a case study and conclude.

2. Related Work

Over the past 20 years, a considerable number of automatic (semantic) annotation systems have been developed. Generally, these annotation systems attach additional data to various concepts, e.g., people, organizations, or places in a given text enriching the documents with machine-processable data. Some famous automatic annotation systems are YEDDA [4], Slate [5], MINTE [6], YAGO [7], or KDTA [8]. For further annotation systems please refer to [9]. Some annotation systems like

OpenCalais [10] even automatically attaches data from DBpedia [11] to extractable named entity (NE) in the text. Generally, the extractable NEs are matched to data in a knowledge bases (KBs) to add additional data from the KB to the document. Some well-known KBs containing common-sense data are DBpedia [11], NELL [12], and KnowledgeVault [13]. Link prediction may identify semantic annotations for an entity, which can be described as the task of estimating the likelihood of a link (relation) existing between nodes (entities), given the links, and attributes of nodes in a graph [14]. Annotation systems aim at developing a graph augmented with common-sense knowledge from external sources. The systems efficiently solve their underlying problem.

However, in this article we investigate a different problem, namely add SCDs to a new document, classify the document to a category, and decide if an agent should extend a reference library with the new document, since automatic annotation systems ignore the context of a reference library.

Surveying methods of text mining, one can base a decision if a new document provides a value for an agent on different aspects, e.g., (i) similarity of text in the spirit of tf.idf [15], comparing a vector representation of a new document with vector representations of the documents in the corpus, (ii) similarity of topics in the spirit of latent Dirichlet allocation (LDA) [16], comparing an estimated topic distribution of a new document with topic distributions of documents in a given corpus, or (iii) entity matching [17] using named-entity recognition (NER), comparing entities (and relations) retrieved from the new document with entities (and relations) from SCDs in the corpus. All three approaches have drawbacks: The first two, based on bag-of-words, ignore SCDs and the order of words. Additionally, the approaches make it difficult to model that a document has to add value, i.e., not be a rephrased copy of an existing document or contain only unrelated data. The last approach has the problem that NER tools might not generate annotations in the context of the task leading to very few matches with SCDs of the corpus. Additionally, the decision in each case is a one-dimensional decision, based on only one feature of the documents. Therefore, we aim at providing an approach to automatically make a multi-dimensional decision that considers the documents' content and context of the task by looking at the individual collection of documents.

Another class of related work deals with HMM-based classification. Classification and statistical learning using HMMs has achieved remarkable progress in the past decades. Using an HMM is a well-investigated stochastic approach for modeling sequential data, and the generation process of HMMs has been successfully applied in a variety of fields, such as speech recognition [18], character recognition [19], finance data prediction [20, 21], credit card fraud detection [22], and workflow mining [23]. Most systems learn an HMM by the Baum-Welch algorithm [24], which is a special case of the EM algorithm [25]. The goal of an HMM is estimating the most likely sequence of hidden states in a dynamic programming fashion by the Viterbi algorithm [1].

3. Preliminaries

This section specifies notations we use in the approach for augmenting and automating corpus enrichment of new documents and gives a brief overview of SCDs.

3.1. *Notation*

We define the following terms to formalize the setting of a corpus containing documents, where each document is associated with a set of additional data, i.e., 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 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 $\#words(d)$ 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 in a corpus represent a reference library for a specific task. The function $V(\mathcal{D})$ returns the corpus-specific vocabulary $\mathcal{V}$ representing the set of all words occurring in the documents of corpus $\mathcal{D}$.
- An SCD t is an RDF triple of the form subject-predicate-object 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 SCDs tuples in corpus $\mathcal{D}$ is then given by $g(\mathcal{D}) = \bigcup_{d \in \mathcal{D}} g(d)$.
- An agent can classify a new document d' based on the documents in corpus $\mathcal{D}$ using the classification function $classify_{\mathcal{D}}(d')$ returning one of the following four categories given document d' and corpus $\mathcal{D}$: (i) *sim* - new document is similar to at least one document in $\mathcal{D}$, (ii) *ext* - new document is an extension of a document in $\mathcal{D}$, (iii) *rev* - new document is a revision of a document in $\mathcal{D}$, and (iv) *unrel* - new document is unrelated to all documents in the corpus. The set $\mathcal{C}$ contains all categories.
- 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 document d surrounding the position ρ in d, i.e., $win_{d,\rho} = (w_{(\rho-i)}^d, ..., w_{\rho}^d, ..., w_{(\rho+i)}^d)$, $i \in \mathbb{N}$ and position ρ marks the middle of the window. The window-specific position of a word $w^d \in win_{d,\rho}$ is given by $pos(w^d, win_{d,\rho})$ (0-based numbering) and the size of window $win_{d,\rho}$ is given by $s(win_{d,\rho}) = 2i + 1$.
- Each word $w^d \in win_{d,\rho}$ is associated with an influence value $I(w^d, win_{d,\rho})$

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

```
1: function buildMatrix(Corpus $\mathcal{D}$)
2:     Input: $\mathcal{D}$
3:     Output: $\delta(\mathcal{D})$
4:     Initialize an $m \times n$ matrix $\delta(\mathcal{D})$ with zeros
5:     for each $d \in \mathcal{D}$ do
6:         for each $t, \rho \in g(d)$ do
7:             for each $w \in win_{d,\rho}$ do                ▷ Iterates over $\rho$
8:                 $\delta(\mathcal{D})[t][w]$ += $I(w, win_{d,\rho})$
9:         Normalize $\delta(\mathcal{D})[t]$                    ▷ See [26]
10:    return $\delta(\mathcal{D})$
```

representing the distance in the text between a word w^d and position ρ. The closer a word w^d is positioned to the position ρ in $win_{d,\rho}$, the higher its corresponding influence value $I(w^d, win_{d,\rho})$. The influence value of a word w^d at $pos(w^d, win_{d,\rho})$ 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 $q = \frac{\rho}{n}$.

3.2. *The Purpose of Subjective Content Descriptions*

Subjective content descriptions are associated with positions in documents and add value to the task of an agent, e.g. SCDs might optimize the performance of a document retrieval service. We generate an additional representation for each SCD by taking a vector of length n, where $n = |V(\mathcal{D})|$ s.t. each vector entry refers to a word in the vocabulary $\mathcal{V}$ of corpus $\mathcal{D}$. 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 $g(\mathcal{D})$. The distribution is represented by an $m \times n$ matrix $\delta(\mathcal{D})$, with the SCD-word distribution vectors forming the rows of the matrix:

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

In Algorithm 1, we fill SCD-word distribution matrix $\delta(\mathcal{D})$ based on the SCDs and words occurring in the documents of corpus $\mathcal{D}$ using a maximum-likelihood strategy such that we count for each SCD t the number of occurrences of each word

Algorithm 2 Estimating MPSCDs

```
1: function MPSCD(Document $d'$, Int $M$, Matrix $\delta(\mathcal{D})$)
2:     Input: $d'$, $M$, $\delta(\mathcal{D})$
3:     Output: $g(d')$, $\mathcal{W}$
4:     $\sigma \leftarrow \frac{\#words(d')}{M}$, $\rho \leftarrow \frac{\sigma}{2}$, $\mathcal{W} \leftarrow \emptyset$
5:     for $\rho \leftarrow \frac{\sigma}{2}$; $\rho \leq \#words(d')$; $\rho += \sigma$ do
6:         Set up $win_{d',\rho}$ of size $\sigma$ around $\rho$ with $t = \perp$
7:         $\delta(win_{d',\rho}) \leftarrow$ new zero-vector of length $n$
8:         for $w \in win_{d',\rho}$ do
9:             $\delta(win_{d',\rho})[w] += I(w, win_{d',t,\rho})$
10:        $t \leftarrow \arg\max_{t_i} \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d',\rho})|}$ in $win_{d',\rho}$
11:        $sim \leftarrow \max_{t_i} \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d',\rho})|}$
12:        $\mathcal{W} \leftarrow \mathcal{W} \cup \{(sim, win_{d',\rho})\}$
13:        $g(d') \leftarrow g(d') \cup \{t\}$
14:    return $g(d')$, $\mathcal{W}$
```

w in the corresponding windows $win_{d,\rho}$ of all documents in $\mathcal{D}$ and all positions. The occurrences are weighted by the influence value of each word in a window. The rows in $\delta(\mathcal{D})$ are normalized to yield a distribution again, i.e., the values of each row sum to 1. Generally, we can calculate the M most probable SCDs for a new document d' given the SCD-word distribution matrix $\delta(\mathcal{D})$ and the sequence of words occurring in d' by partitioning d' into M segments s.t. each segment represents one window. We compute the M MPSCDs for d' by building a vector representation $\delta(win_{d',\rho})$ for each of the M windows using the words in the M windows of d' and estimate the SCD from $\delta(\mathcal{D})$ that has a vector representation most similar to $\delta(win_{d',\rho})$. The SCD t most similar to $\delta(win_{d',\rho})$ (w.r.t. the cosine similarity) is given by:

$$\arg\max_{t} \frac{\delta(\mathcal{D})[t] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[t]| \cdot |\delta(win_{d',\rho})|}. \quad (2)$$

Algorithm 2 describes the process of estimating MPSCDs for document d' using the corpus-specific SCD-word distribution $\delta(\mathcal{D})$ after specifying the number of SCDs (M) an agent is interested in. The output of Algorithm 2 includes the set of MPSCDs $g(d')$, one for each of the M windows, as well as a set of tuples of similarity value and window for each MPSCD. For further details, we refer to [26].

4. Automatic Corpus Enrichment

This section presents a new approach for augmenting and automatically corpus enrichment of a new document while considering the context given by the documents in a corpus. We introduce the problem of classifying a new document d' given an individual composition of documents in corpus $\mathcal{D}$, and we present an HMM-based approach to identify the category of the new document d' by estimating the probability of the most likely sequence of hidden states from the observable MPSCD similarity values in document d'. We then provide a decision making procedure and describe how an agent can use the most likely sequence to augment corpus enrichment with providing location-specific information about new data within the context of the specific corpus. Even though one might use any identified category relevant to a specific task, we consider the four document categories *sim*, *ext*, *rev*, and *unrel*. Before going into details about document classification techniques, we first describe each document category in the next section and take a look at the output of Algorithm 2, specifically at the similarity values to show how document classification has a chance at success using the sequence of similarity values.

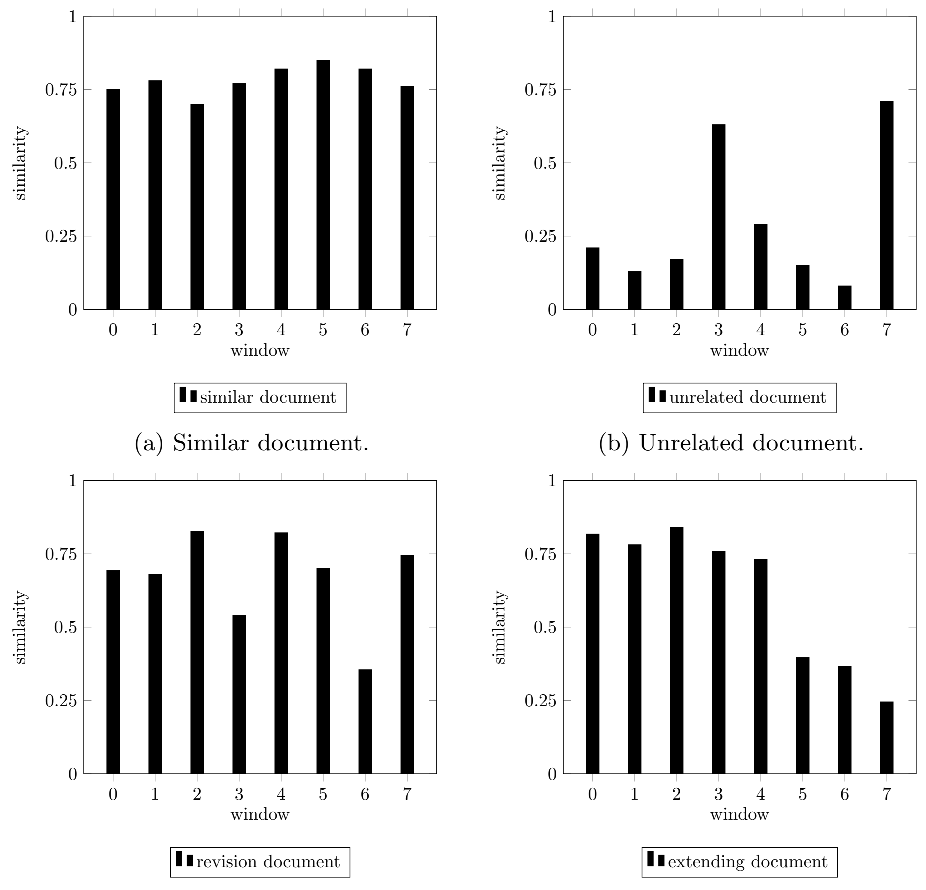
4.1. *Document Categories and their Similarity Values*

We assume that a document is labelled with a document category and we follow the idea of [26] that the category of a document can be detected by considering the MPSCD similarity values. Before presenting details about the document categories, let us consider an example of all four categories.

Example 1. Assume that we have four documents of similar length and each of them from one of the four categories *sim*, *ext*, *rev*, and *unrel*, respectively. Figure 1 presents the similarity value characteristics of the MPSCDs for each of the four documents and we use seven windows to segment the content within the documents. The x-axis represents the SCD window number within each document, and the values on the y-axis represent the similarity value of the MPSCD. Generally, the similarity values change slightly between neighbouring windows for both similar and unrelated documents and only few major shifts between neighbouring windows are given. But, the MPSCD values of similar documents are considerably higher compared to unrelated documents. Extending documents are characterized by high similarity values in the initiating SCD windows and smaller similarity values in the later windows. The similarity value for the windows' MPSCD of revision documents behave like similar documents. However, there exists windows having a significant change in the similarity value of the neighbouring windows, namely, those windows having new content which is unrelated in the context of the corpus.

Based on the observations shown in the last example, we describe the document categories more detailed.

Similar documents (*sim*). The content of a new document d' is classified as *sim* if the document's content is related to the content of a subset of documents in corpus $\mathcal{D}$, i.e., the new document tells about the same event, same persons, or same topics. Thus, values in the MPSCD similarity sequence are mostly high and contain only few entries with slightly lower values, i.e., the range of MPSCD similarity values is small. A new document d' is a similar document w.r.t. the documents in a corpus



(a) Similar document. (b) Unrelated document.

(c) Revision document. (d) Extending document.

Fig. 1: Representation of MPSCD similarity value characteristics for a similar (a), unrelated (b), revision (c), and extending (d) document in an exemplary way.

$\mathcal{D}$ if

$$\forall win_{d',\rho} \in d' : \max_{t_i} \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d',\rho})|} > th_1. \tag{3}$$

Threshold th_1 describes the minimum required relatedness for each window. The threshold is subjective and depends on the individual composition of documents in $\mathcal{D}$. We use $th_1 = 0.7$.

Unrelated documents (*unrel*). The content of a new document d' is classified as *unrel* if the document's content is unrelated to the content of all documents in $\mathcal{D}$. The values in the MPSCD similarity sequence of d' are mostly low and only a small number of windows contain higher similarity values. Thus, we define a new document d' as an unrelated document w.r.t. the documents in a corpus $\mathcal{D}$ if

$$\forall win_{d',\rho} \in d' : \max_{t_i} \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d',\rho})|} < th_2. \tag{4}$$

Threshold th_2 describes the maximum relatedness for each window in document d'. The threshold is subjective and depends on the individual composition of documents in $\mathcal{D}$. We use $th_2 = 0.3$.

Extensions (*ext*). The content of a new document d', representing an extension of another document $d \in \mathcal{D}$ is mostly related to the content of at least one document in $\mathcal{D}$ but additionally contains content which is not available in documents of $\mathcal{D}$, i.e., the new document is an extended version of another document in corpus $\mathcal{D}$. Thus, new document d' can be represented by $d' = [a, b]$, where a represents the first windows of d' containing MPSCD similarity values greater than th_1 and b represents the second part of windows having MPSCD similarity values smaller than th_2. We can represent window property using the following regular expression: a^+b^+.

Revisions (*rev*). The content of a new document d', representing an revision of another document $d \in \mathcal{D}$ is generated by *appending*, *replacing*, or *removing* some sentences of document d. The MPSCD similarity value of most windows is greater than th_1 and the MPSCD similarity value of few windows is less than th_2. We can represent the described behaviour as the following regular expression: $(a^+b \mid b^+a)(a \mid b)^*$. Again, a and b represent values greater than th_1 and smaller than th_2, respectively.

Next, we define the problem of classifying a new document and present an approach to solve the problem using an ensemble of four HMMs.

4.2. *Document Classification Problem*

After estimating the MPSCDs for a new document d' using Algorithm 2, we can extract a sequence of MPSCD similarity values over the SCD-windows in d'. The document classification problem asks for the most probable category of document d' given the observable sequence of MPSCD similarity values and documents in $\mathcal{D}$. More technically, we can represent the classification problem

$$\arg\max_{c \in \mathcal{C}} P(c|\mathcal{W}). \tag{5}$$

As described in the previous section, we follow the idea that if the content in a window of a new document d' is (un)related to the agent's task, then (low) high similarity values occur, which may vary over the course of a new document, with known and unknown parts mixing. Generally, we can only look at the observable MPSCD similarity values, but cannot directly observe if the content in a window is (un)related to the task of an agent. The consequence of this consideration is that we have a sequence of hidden states, encoding the relatedness of the document's

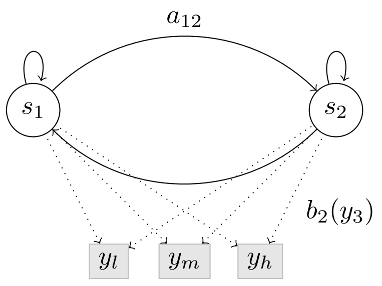


Fig. 2: HMM – hidden states $\{s_1, s_2\}$ emitting observation symbols $\{y_l, y_m, y_h\}$.

content, and a sequence of observations given by the similarity values of MPSCDs. Together, we can model the described setup as an hidden Markov model. We are interested in solving the document classification problem by learning four HMMs, one for each category of documents, determining the most likely sequence of hidden states in the new document d'. Finally, we return the category associated with the HMM having the highest probability of hidden state sequence. Calculating the most likely sequence of hidden states allows us to analyze documents over “time”, i.e., sentence by sentence. Additionally, we can consider the sequence of the MPSCD similarity values, instead of simply using statistic values like the maximum or minimum in a set of MPSCD similarity values proposed in [26]. Generally, the problem of estimating the most likely sequence of hidden states is often associated with hidden Markov models.

4.3. *An Ensemble of HMMs*

We form an ensemble of HMMs to detect category of a new document.

Definition 1 (Hidden Markov model). An HMM $\lambda = (a_{ij}, b_j, \pi)$ for classifying documents of some category consists of

- a set of hidden states given by $\Omega = \{s_1, ..., s_n\}$, where $n = 2$, with state s_1 representing related content and s_2 representing unrelated content,
- an observation alphabet $\Delta = \{y_1, \ldots, y_m\}$, where each observation symbol represents a range of MPSCD similarity values,
- a transition probability matrix A representing the probability between all possible state transitions $a_{i,j}$ of the two hidden 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 hidden state in Ω, and
- an initial state distribution vector $\pi = \pi_0$.

Figure 2 contains a graphical representation HMM $\lambda = (a_{ij}, b_j, \pi)$ containing observation symbols $\{y_l, y_m, y_h\} \in \Delta$, and the two hidden states $\{s_1, s_2\} \in \Omega$.

Definition 2 (Transition Probability). 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

$$a_{i,j} = P(s_j|s_i), \tag{6}$$

and represented by the following state transition matrix A:

$$A = \begin{array}{c} \\ s_1 \\ s_2 \\ \vdots \\ s_n \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & \cdots & s_n \end{array} \\ \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \end{array}$$

Definition 3 (Emission Probability). The entries of emission probability matrix B represent the probability to emit symbol $y_k \in \Delta$ in hidden state $s_j \in \Omega$ and, with $\sum_{j=1}^{m} b_j(y_k) = 1$, are given by

$$b_j(y_k) = P(y_k|s_j). \tag{7}$$

The semantics of λ is given by unrolling λ for a given number of slices and building a full joint distribution. For a set of document categories $\mathcal{C}$, an ensemble of HMMs $\mathcal{H}$ contains an HMM for each document category in $\mathcal{C}$. All $\lambda \in \mathcal{H}$ have the same Ω and Δ but different A and B. We represent the emission probability between observations and hidden states using the following observation matrix B:

$$B = \begin{array}{c} \\ s_1 \\ s_2 \\ \vdots \\ s_n \end{array} \begin{array}{c} \begin{array}{cccc} y_1 & y_2 & \cdots & y_m \end{array} \\ \begin{pmatrix} b_1(y_1) & b_1(y_2) & \cdots & b_1(y_m) \\ b_2(y_1) & b_2(y_2) & \cdots & b_2(y_m) \\ \vdots & \vdots & \vdots & \vdots \\ b_n(y_1) & b_n(y_2) & \cdots & b_n(y_m) \end{pmatrix} \end{array}$$

Generally, for a corpus $\mathcal{D}$, both, the transition probability matrix A and the emission probability matrix B for each document category specific HMM is unknown. One famous technique for learning both matrices of an HMM is the Baum-Welch algorithm [24], which is a special case of the well-known EM algorithm [25]. Using a set of documents where each document is from a specific category $c \in \mathcal{C}$, one can calculate for a hold-out set the MPSCDs and the corresponding MPSCD similarity values s.t. we can train one HMM for each category on this data using the Baum-Welch algorithm.

The observation alphabet Δ requires discretizing similarity values. As a function $f : [0,1] \mapsto \Delta$, it maps a similarity value x to one of the m symbols in Δ:

$$f(x) = \begin{cases} y_1 & 0 \leq x < th_1 \\ y_2 & th_1 \leq y < th_2 \\ \vdots & \\ y_m & th_{m-1} \leq y < 1 \end{cases} \tag{8}$$

Algorithm 3 MPSCD Similarity Discretization

```
1: function DISCRETIZE($\mathcal{W}$, $f$)
2:     Input: $\mathcal{W}$, $f$
3:     Output: $O$
4:     $O \leftarrow ()$                          ▷ observation sequence
5:     for each $sim \in \mathcal{W}$ do
6:         $y_k \leftarrow f(sim)$                ▷ $y_k \in \Delta$
7:         $O \leftarrow O \circ y_k$
8:     return $O$
```

Algorithm 3 shows pseudocode, mapping the similarity values in $\mathcal{W}$ for a new document d' to the observation symbols in alphabet Δ based on function f, returning a sequence of observable symbols O. Generally, the discretization and its thresholds depend on the task of an agent and can be adapted to each problem individually. We follow the idea of Kuhr et al. [26], using $m = 3$, and select $th_1 = 0.3$ and $th_2 = 0.7$ such that $f(x)$ maps MPSCD values to y_l, y_m, and y_h, referring to low, medium, and high values.

Next, we present an approach for estimating the most probable category of a new document by providing the sequence of observable MPSCD similarity values to an ensemble of corpus- and category-specific HMMs.

4.4. *Detecting the Category of a New Document*

To solve the document classification problem, we have to identify the most likely sequence of hidden states from alphabet Ω, given a sequence of observation symbols from alphabet Δ, in each HMM of the ensemble of category-specific HMMs. Each of the most likely sequences is associated with a probability. The document category is then given by the category for which the HMM with the highest probability for its sequence has been learned.

The task of determining which sequence of variables is the underlying source of some sequence of observations is called a *decoding task*. We calculate for each HMM the most likely sequence of hidden states using the Viterbi algorithm [1] which makes use of the dynamic programming trellis for computing the most likely hidden state sequence S for an observation sequence O. Before presenting a complete specification of the document classification approach, let us consider an example of a most likely sequence given a set of MPSCD similarity values.

Example 2. Assume that we have calculated a set of MPSCDs for a new document d' by using Algorithm 2 and the corresponding MPSCD similarity values are as follows:

$$(0.29, 0.41, 0.59, 0.48)$$

Algorithm 4 Document Classification

```
1:  function docCategoryDetection($\mathcal{W}$, $\mathcal{H}$, $f$)
2:      Input: $\mathcal{W}$, $\mathcal{H}$, $f$
3:      Output: $s$
4:      $p \leftarrow 0$                                   ▷ current highest probability
5:      $s \leftarrow$ initialize                          ▷ output tuple
6:      $O \leftarrow$ discretize($\mathcal{W}$, $f$)      ▷ observation sequence, see Alg.3
7:      for each $\lambda \in \mathcal{H}$ do
8:          $S \leftarrow$ Viterbi($\lambda, O$)
9:          $y \leftarrow prob(S)$                         ▷ probability of $S$
10:         if $y > p$ then
11:             $p \leftarrow y$
12:             $s \leftarrow (S, \lambda)$
13:     return $s$
```

Using Algorithm 3 leads to the following observation sequence:

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

Assume that the hidden state sequence for a specific configuration of the transition and emission probability matrices of an HMM is given by

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

Figure 3 represents the trellis of the observation sequence O_1, where the thick arrows indicate the most probable transitions between the hidden states and the dotted lines represent all possible hidden state transitions.

Algorithm 4 describes how to estimate the most probable category of a new document using an ensemble of HMMs. The input parameters are given by the MP-SCD similarity values in $\mathcal{W}$ (output of Algorithm 2) and the ensemble $\mathcal{H}$. In line 2 and line 3, Algorithm 4 initializes a temporary variable p and the output tuple s.

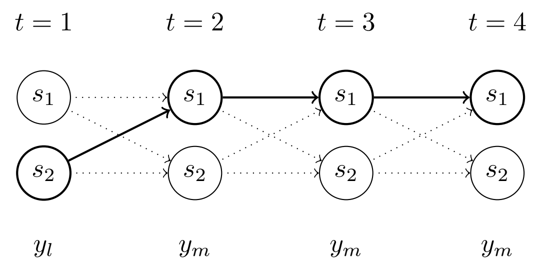


Fig. 3: Trellis of $O = (y_l, y_m, y_m, y_m)$, leading to the following most likely sequence of hidden states: (s_2, s_1, s_1, s_1).

In line 4, Algorithm 4 calls Algorithm 3 to set the observation sequence O to the sequence of the similarity values in $\mathcal{W}$ discretized based on function f as in Equation 8. Afterwards, Algorithm 4 iterates over the given ensemble of HMMs in $\mathcal{H}$. In each iteration, the algorithm calculates a most probable sequence S given observables O using the Viterbi algorithm (line 6). In line 8 to line 10, Algorithm 4 tests if the probability $prob(S)$ of the current most likely sequence S is higher than the previously seen highest probability. If the probability is higher, Algorithm 4 saves the current most likely sequence S and the HMM λ in variable s. After iterating over all HMMs in $\mathcal{H}$, Algorithm 4 terminates and returns the most likely sequence s that has exhibited the highest probability among all most likely sequences as well as the corresponding HMM. This output we then use as a basis for the decision regarding corpus enrichment. Before moving on to decision making, let us consider an example for how document categories and sequences may interact.

Example 3. Assume that we have an unrelated document with similarity values that have led to the following observation sequence as well as the most likely sequence as an output of Algorithm 4, meaning the HMM learned on unrelated documents has produced the most likely sequence with the highest probability:

$$O_{unrel} = (y_l, y_m, y_l, y_l) \tag{9}$$

$$S_{unrel} = (s_2, s_2, s_2, s_2) \tag{10}$$

With observations of y_l mainly, the most likely sequences of the other HMMs have a much lower probability as the evidence for unrelated content is very high, which is associated with low probabilities in them. Given, e.g., an extension of a document, the observation sequence may look as follows with the HMM learned on extensions yielding the most likely sequence with highest probability also given:

$$O_{ext} = (y_h, y_m, y_l, y_l) \tag{11}$$

$$S_{ext} = (s_1, s_1, s_2, s_2) \tag{12}$$

Here, the HMM trained on unrelated documents can only explain the last part with high probability whereas the HMM trained on extensions can explain both parts.

Estimating the category of a new document d' can help an agent making a decision on extending a corpus with the new document d' or not, e.g., if an agent is interested in new documents containing similar content to documents in the corpus the agent can simply add new documents classified as similar document.

4.5. *Automatic Decision Making: Corpus Enrichment*

Given corpus $\mathcal{D}$ and the corresponding set of SCDs $g(\mathcal{D})$, an agent has to perform following steps *offline*:

(i) Build an SCD-word matrix $\delta(\mathcal{D})$ based on $g(\mathcal{D})$.
(ii) Train an ensemble of HMMs $\mathcal{H}$ using $\mathcal{D}$ and $\delta(\mathcal{D})$.

(iii) Specify function f for discretizing similarity values.
(iv) Specify which document category are relevant for the task and the current condition of corpus $\mathcal{D}$.

The steps highly depend on the task at hand and differ from one setting to the next. Relevant document category may be extensions of documents for an agent to extend its own knowledge base. If a corpus is already very large, an agent may be interested in updating its documents with newest revisions. If a task changes and documents not longer appear to help in solving its task, unrelated documents may provide the most added value (without any further information about the agent's information need). As just indicated, specifications may even change over time when a corpus grows and a need for specific documents becomes apparent. Repeating the steps after many changes in a corpus to update $\delta(\mathcal{D})$ or $\mathcal{H}$ may become prudent.

Faced with a new document d', an agent performs the following steps *online*:

(i) Use Algorithm 2 to estimate the MPSCDs for document d'.
(ii) Use Algorithm 4 to estimate the category of d'.
(iii) Based on document's category and its own specification, add/forgo d'.

Basing the decision on most likely sequences enables an agent to further process the most likely sequence with the highest probability and analyze documents over "time", i.e., sentence by sentence. As such, this sequence allows for augmenting corpus enrichment by being able to pinpoint interesting positions in a document, which we consider next.

4.6. *Augmenting Enrichment: Positions of Interest*

Let us assume that an agent is interested in extending a corpus $\mathcal{D}$ with new documents and that the documents are classified as unrelated documents but share at least one section of related content with one other document. The output of document classification Algorithm 4 includes the most likely sequence of hidden states, representing for each SCD window if the content is related or unrelated to the content of documents in corpus $\mathcal{D}$. We can use the sequence of hidden states to extract information about the content similarity in an SCD window instead of only having information about the similarity values of the most probable SCD relating to a specific window. If a new document d' is unrelated to other documents in corpus $\mathcal{D}$ but parts of the document contain related content, then the agent might be interested in the parts containing related content to identify if document d' might contain new content relating to an already-known topic. We describe these parts in a new document d' as positions of interests.

Identifying positions of interests (PoIs) in a new document d' requires detecting the category of d' using Algorithm 4. Depending on the category of the document d', an agent might be interested in different positions within document d'. Thus, we specify the following positions of interest depending on the document's category. (i) For documents in category *sim*, sections containing unrelated content represent

PoIs, (ii) documents of in category *dext* contain related content in the beginning of the document, which changes to unrelated content when the document extension starts. The position where the similarity values changes represents the PoI in *ext*, (iii) for document of in category *rev*, the PoI is given by those positions containing modified (unrelated) content, and (iv) for documents in category *unrel*, the sections containing related content represent PoIs.

5. Optimizing Subjective Content Descriptions

In the last section, we have introduced an approach for classifying a new document into one of four categories. The document classification approach is based on an initial set of SCDs occurring in documents of corpus $\mathcal{D}$ as well as the estimated MPSCD and the corresponding similarity value for each window of the new document. In this section, we present two additional approaches for optimizing the quality of initially estimated MPSCDs. The first approach contains an adaptation technique adapting the size and position of SCD windows. Second approach adapts the initial SCD-word distribution specifically for documents we are interested in extending the corpus with.

5.1. *Adapting Subjective Content Description Windows*

Kuhr et al.[26] have introduced Algorithm 2. The algorithm returns an initial set of located MPSCDs $g(d')$ for a corpus-extending document d' as well as a set of tuples $\mathcal{W}$ containing the similarity values sim of the MPSCDs in $g(d')$ and the corresponding windows $win_{d',\rho}$. Both, the initial located MPSCDs $g(d')$ and the set of tuples $\mathcal{W}$ are based on SCD-word distribution $\delta(\mathcal{D})$ and a uniformly distributed set of equally sized windows. We argue that the MPSCDs in $g(d')$ represent only a first estimate for d' because the position and size of the corresponding SCD windows might be non-optimal. If an agent is interested in extending its corpus with a document, than the agent benefits from optimized content descriptions.

We aim to optimize the MPSCDs in $g(d')$ by iteratively adjusting the size and position of all windows in a way that the overall similarity values of the M MPSCDs is at maximum by solving the following problem,

$$\max \sum_{(sim, win_{d',\rho}) \in \mathcal{W}} sim. \tag{13}$$

Thus, we describe an iterative MPSCD optimization approach in Algorithm 5 improving the initial MPSCDs similarity values for document d', such that the overall MPSCD-similarity value increases. Algorithm 5 requires the following input parameters: (i) new document d', (ii) number M of MPSCD, (iii) initial set of windows and corresponding cosine similarity ($\mathcal{W}$), (iv) SCD-word distribution $\delta(\mathcal{C}_{d'})$, and (v) initial set of MPSCD $g(d')$. Finally, Algorithm 5 returns $g(d')$ containing an optimized set of located SCDs for document d'.

Algorithm 5 MPSCD Optimization

```
1:  Input: $d'$, $M$, $\delta(\mathcal{C}_{d'})$, $\mathcal{W}$, $g(d')$
2:  Define: $t^*$, $sim'$, $sim^*$, $\rho$, $winT_{d',\rho'}$, $sum, sum'$
3:  Output: $g(d')$
4:  $sum \leftarrow \sum_{(sim,win_{d',\rho})\in\mathcal{W}} sim$
5:  $sum' \leftarrow sum$
6:  while $sum' \geq sum$ do
7:      $sum \leftarrow sum'$
8:      for each $(sim, win_{d',\rho}) \in \mathcal{W}$ do
9:          $sim^* \leftarrow sim$
10:         for each window adjustment $a \in \{l^+, r^+, l^-, r^-\}$ do
11:             $winT_{d',\rho'} \leftarrow win_{d',\rho}$
12:             Update $winT_{d',\rho'}$ based on $a$
13:             $sim' \leftarrow \max_{t_i} \frac{\delta(\mathcal{C}_{d'})[i]\cdot\delta(winT_{d',\rho'})}{|\delta(\mathcal{C}_{d'})[i]|\cdot|\delta(winT_{d',\rho'})|}$
14:             if $sim' > sim^*$ then
15:                 $t^* \leftarrow \arg\max_{t_i} \frac{\delta(\mathcal{C}_{d'})[i]\cdot\delta(winT_{d',\rho'})}{|\delta(\mathcal{C}_{d'})[i]|\cdot|\delta(winT_{d',\rho'})|}$
16:                 Update $win_{d',\rho}$ with $t^*$ and $\rho'$
17:                 $sim^* \leftarrow sim', sim \leftarrow sim'$
18:                 $g(d') \leftarrow g(d') \setminus \{t\} \cup \{t^*\}$
19:     $sum' \leftarrow \sum_{(sim,win_{d',\rho})\in\mathcal{W}} sim$
20: return $g(d')$
```

In line 4 of Algorithm 5, we calculate the overall cosine similarity of the initial MPSCDs in $g(d')$ using the similarities stored in $\mathcal{W}$. Afterwards, in line 5 to line 18, we iteratively adjust the SCD windows trying to obtain a higher cosine similarity between the adjusted influence vector and the SCD vector in $\delta(\mathcal{C}_{d'})$. We perform the window adjustments until we are reaching a local optimum for the optimization problem stated in 13. We can adjust each window in $\mathcal{W}$ in the following four different directions: (i) extend left boundary of the SCD-window to the left side (l^+), (ii) extend right boundary of SCD-window to the right side (r^+), (iii) shift left boundary of SCD-window to the right side (l^-), and (iv) shift right boundary of SCD-window to the left side (r^-). The first two window adjustments extend the size of $win_{d',\rho}$, while the last two window adjustments reduce the size of $win_{d',\rho}$. We calculate for each possible window adjustment the MPSCDs and choose the adjustment, for which the overall similarity is maximized.

Theorem 1. *Algorithm 5 estimates for a corpus-extending document d' the M (locally) most probable SCDs and finally terminates.*

Proof. Algorithm 5 optimizes the initial M SCDs in $\mathcal{W}$ by iteratively adjusting the M windows in a fashion s.t. the overall similarity between the influence vectors of all SCD windows and the MPSCD is maximized for the new document d'. The resulting MPSCDs in $g(d')$ depend on the initial position of SCD windows, since we consider in the iterative optimization only window adjustments of size one and do not change the number of initially defined SCD windows (M). Thus, the final MPSCDs represent a (local) optimum for the M SCD windows. Finally, Algorithm 5 terminates, since the number of possible window adjustments in the algorithm are finite. □

5.2. *Augmenting Enrichment: Document Clusters*

If an agent is interested in extending its corpus $\mathcal{D}$ with a document d' containing similar content to documents in $\mathcal{D}$, we can adjust the process for generating SCD for d' leading to good SCD. Then, the first step in estimating most probable SCDs for a corpus-extending document d' consists of identifying only documents in corpus $\mathcal{D}$ having a high topic similarity with d'. We form a cluster $\mathcal{C}_{d'}$ of d'-related documents such that all documents $d \in \mathcal{C}_{d'}$ have a Hellinger distance H of their topic distributions $\theta_d, \theta_{d'}$ being smaller than a threshold τ, i.e.,

$$\mathcal{C}_{d'} = \{d \in \mathcal{D} \mid H(\theta_{d'}, \theta_d) < \tau\}. \tag{14}$$

The threshold τ decides on the minimum required similarity between two documents d' and d, such that document $d \in \mathcal{C}_{d'}$. The best value for τ depends on the performance of external applications working with the located SCDs in $g(d')$. The topic distribution of documents in $\mathcal{D}$ changes with respect to α, β, and K in the LDA. The smaller τ is, the higher the topic similarity between the documents in $\mathcal{C}_{d'}$ is. Equation 14 requires a topic distribution $\theta_{d'}$ for the corpus-extending document d' as well as the documents in $\mathcal{D}$. There exist three basic strategies to arrive at the topic distributions, (i) extend corpus $\mathcal{D}$ with new document d' and generate a new topic model from $\mathcal{D} \cup \{d'\}$, which contains topic distributions θ for each document in $\mathcal{D} \cup \{d'\}$, (ii) generate a topic model from $\mathcal{D}$, which contains topic distributions θ for each document in $\mathcal{D}$, and approximate the topic distribution $\theta_{d'}$ using *folding in* Gibbs sampling [27], or (iii) generate a topic model from $\mathcal{D}$, which contains topic distributions θ for each document in $\mathcal{D}$, and approximate the topic distribution $\theta_{d'}$ using *folding in* Gibbs sampling as in (ii), while also updating the overall topic model, known as the online variational Bayes algorithm for LDA [28]. Each strategy has its advantages and disadvantages. We refer to [29] for further details about the different techniques and their advantages and disadvantages. However, whichever strategy we use for learning topic distributions, afterwards we can use the Hellinger distance between d' and all documents in $\mathcal{D}$ to assemble $\mathcal{C}_{d'}$.

Now, we can perform Algorithm 1 with $\mathcal{C}_{d'}$ as input. This leads to a new SCD-word distribution $\delta(\mathcal{C}_d)$ which we use as input for Algorithm 2 calculating a new set of SCDs for document d'. We evaluate the performance using document clusters in the second part of the next section.

6. Case Study

After introducing the document classification approach as well as some optimization techniques, we present a case study illustrating the potential of document classification using the sequence of observable MPSCD similarity values as evidence in HMMs. We use the following three corpora within the case study, each containing articles from the free Wikipedia encyclopedia: (i) Cities in Europe: 36 documents about the largest cities in Europe (https://bit.ly/2kOvmwD). (ii) US presidents: 45 documents about presidents of the U.S. between 1789 and 2017 (https://bit.ly/2Z1v1G9), and (iii) US universities: subset of documents 50 about the state and territorial universities in the U.S. (https://bit.ly/2mOzXAW).

6.1. *Preprocessing*

We work with a Java-based implementation of Algorithm 2 and Algorithm 5 and use their results as input parameters in our Python-based implementation of Algorithm 3 and Algorithm 4. Initially, we download all necessary articles from Wikipedia for the three corpora. We preprocess the articles to generate a topic model by: (i) lowercasing all characters, (ii) stemming the words, (iii) tokenizing the result, and (iv) eliminating tokens from a stop-word list of 337 words. For Algorithm 2 and Algorithm 5 we only remove stop-words from the documents. Then, we generate for each category of a document in each of the three corpora a training set and test set in the following way: (i) Similar Documents (*sim*): We choose 90% of the corresponding Wikipedia articles as training data, i.e., 90% of the articles represent documents in the corpus. The remaining 10% of articles from the same corpus are used as a held-out set acting as the test data. (ii) Extending documents (*ext*): For each corpus, we choose the training set by selecting the corresponding Wikipedia articles and extend them with text from articles of the other two corpora, such that the document extending content is unrelated to the content in the current corpus. We proceed the same way to generate documents for the test set. (iii) Revision documents (*rev*): We use the *view history* function in Wikipedia generating a revised version of a document. The latest version of an article contains modified sentences and additional content, since the latest article represents a revision of an older version of the same article. Again, we proceed the same way for generating the document in the test set. (iv) Unrelated documents (*unrel*): We randomly choose the training set by selecting 90% of articles from one corpus and randomly selecting articles from the two other corpora to generate a test set.

For each category of document, we take its training set and learn the transition and emission function for the category-specific HMMs using the Baum-Welch algorithm. This process results in four HMMs each of them representing the configuration for one category of document for each corpus. We use Algorithm 4 to perform the document classification of documents in the test set. Precision and recall represent the performance of Algorithm 4, where true positives (tp) refer to the number of documents whose document category have been correctly estimated,

Table 1: Document classification performance considering the four document categories *sim*, *unrel*, *ext*, *rev*, and the three corpora *city*, *university*, *president*.

Corpus	Method	Document category			
		sim	*unrel*	*ext*	*rev*
City	Precision	0.72	1.0	0.93	0.70
	Recall	0.65	1.0	0.86	0.41
	F1-Score	0.68	1.0	0.89	0.52
University	Precision	0.77	1.0	0.91	0.72
	Recall	0.71	0.96	0.84	0.58
	F1-Score	0.72	0.98	0.87	0.64
President	Precision	0.70	1.0	0.86	0.68
	Recall	0.71	1.0	0.92	0.51
	F1-Score	0.70	1.0	0.89	0.58

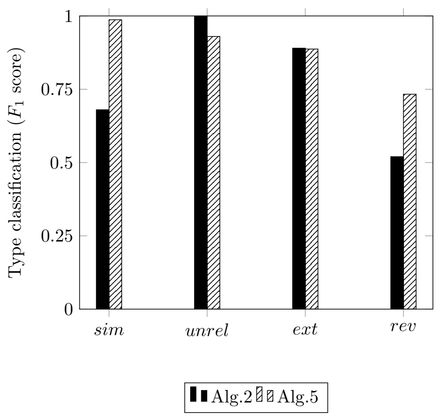
false positives (fp) refer to the number of documents whose document category have been falsely estimated, and false negative (fn) to the number of documents classified with a category of a document that was not found.

6.2. *Document Classification*

We conduct experiments to test the performance of the document classification approach based on the earlier described technique to generate corpora and the evaluation approach. Table 1 represents the performance of Algorithm 4 using both, precision and recall for all three corpora. Generally, document classification performs best on unrelated and extending documents. The transition probability and emission probability are similar for categories *sim* and *rev* for documents in the corresponding HMMs. We have never classified revisions and similar documents as unrelated documents or document extensions, respectively, but sometimes we have classified documents from category *sim* as documents from category *rev*, and vice versa. We assume an agent interested in similar documents might accept document revisions because they are similar to the documents in a corpus. If an agent is only interested in similar documents or revisions, it must analyse documents of both categories in detail, e.g., by using positions of interest. Kuhr et al. [26] have had the same problem in distinguishing a new document from category *sim* and *rev*. Their introduced indicators were the same for similar and revised documents in the president corpus, and only one of the five indicators is different for both categories in the city corpus. Next, we look at the performance of the optimization techniques.

6.3. *Window Adaptation*

Estimating the MPSCDs for new documents requires a predefined number (M) of MPSCDs we are interested in as well as the size of the corresponding MPSCD-windows. In Section 3, we have described Algorithm 2 dividing document d' into M



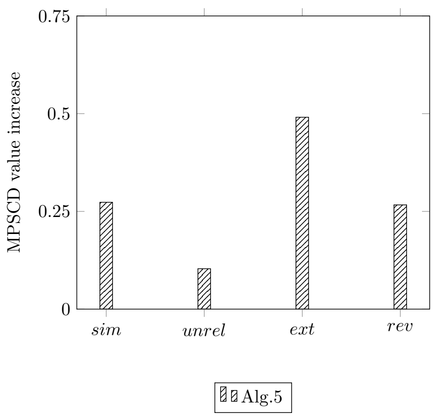
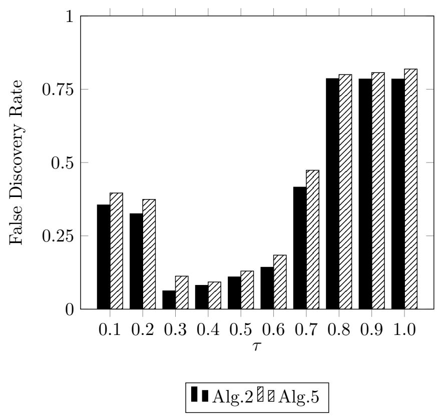


Fig. 4: Left: Classification performance using Alg. 2 and Alg. 5. Right: Average MPSCD increase for each document after optimizing the window size using Alg. 5.

equally sized windows (line 4). Afterwards, it estimates for each of the M windows the MPSCD using SCD-word distribution $\delta(\mathcal{D})$. Obviously, the initial segmentation of document d' ignores the content within documents. Thus, we have introduced Algorithm 5 adapting the windows' size to increase the quality of SCDs causing an optimization of the document classification performance. Left plot in Figure 4 represents the performance of Algorithm 5 by comparing the document classification performance based on Algorithm 2 and Algorithm 5 for documents in the city corpus. The overall classification performance of Algorithm 5 is better than Algorithm 2 and the HMM generated from output of Algorithm 5 lead to markable increase in detecting documents of category *sim* and *rev*. Right plot in Figure 4 represents the average MPSCD similarity value increase for a document after optimizing the window size using Algorithm 5. Adapting one window will automatically adapt the neighbouring windows and might result in a decreasing MPSCD similarity value for the neighbouring windows. Generally, Algorithm 5 adapts the windows in a way that the sum of all MPSCD similarity values increases and the document category classification performance increases, too. The false discovery rate (FDR) slightly increase using Algorithm 5, since the algorithm focus on the optimization of overall MPSCD similarity value increase instead of optimizing the FDR.

6.4. *Document Clusters*

We use the similarity values of MPSCDs to identify the category of new documents. Let us assume that an agent is interested in enriching its initial corpus only with similar documents. Than, the agent can use similarity values of MPSCDs to identify the category of a new document on the documents in its corpus. Classifying new documents based on the MPSCDs similarity values is great, since an agent can reuse the MPSCD corresponding to the estimated similarity value. This means an agent



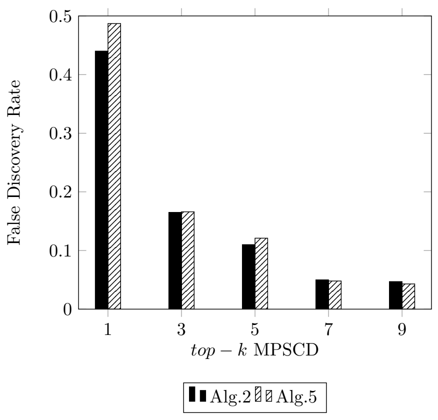


Fig. 5: Left: FDR of MPSCDs for new documents of category sim contemplating only documents from the corpus having a topically Hellinger distance of τ. Right: FDR of MPSCDs for similar new documents contemplating the top-k MPSCDs.

has directly corpus-specific content description for a corpus-extending document. However, generally a new document is only eminently related to a subset of documents within a large corpus. Therefore, we are interested in analysing the content description performance for a new document d' using only a subset of documents $\mathcal{C}_{d'}$ from corpus $\mathcal{D}$ containing documents having a similar topic-distribution. We use the FDR analysing the performance of calculated MPSCDs which is defined as $FDR = FP/(FP + TP)$. In Figure 5, we present the FDR of the MPSCDs corresponding to the MPSCD similarity values used to classify documents. Left plot shows FDR of estimated MPSCDs for documents of category sim using the Hellinger distance to estimate the documents in $\mathcal{C}_{d'}$, having a distance less than τ. The FDR is best contemplating only documents having a Hellinger distance between 0.3 and 0.6. Obviously, ignoring a subset of documents from $\mathcal{D}$ changes the SCD-word distribution $\delta(\mathcal{D})$. However, contemplating only documents having a very low distance ignore the fact that we are approximating the topic distribution for new documents and documents containing relevant SCDs might not in $\mathcal{C}_{d'}$, since their Hellinger distance is higher than the specified value for τ. The right plot shows the FDR of MPSCDs for documents of category sim, too. We use the average FDR with values 0.3, 0.4, and 0.5 for τ to analyse the impact of different $top - k$ MPSCD setting.

7. Conclusion

If an agent is presented with a new document this article enables to decide whether to extend its corpus with the new document or not by classifying the document in a context-specific way. We present how to model a window sequence with hidden Markov models estimating the most probable sequence of (un)related content in a document to estimate the category of a new document based on the documents in a corpus. The HMMs form an ensemble that can be easily extended with HMMs for other document categories or weighted to prioritise documents of a certain category. As classifying documents is carried out using the most likely sequence in the HMMs, we can use this sequence to pinpoint locations in documents that may contain new content. Additionally, we have introduced two optimization techniques, namely, window adaptation and document clusters. Using document clusters an agent can generate an MPSCD-word distribution matrix δ resulting in high-quality MPSCDs for new documents extending to the corpus. Furthermore, an agent can adapt the MPSCD-similarity values using an window optimization algorithm to increase the document classification performance for new documents. However, the parameters in each of the category specific HMMs change for each corpus and it is difficult to reuse category specific HMMs from one corpus for document in another corpus. Therefore, our future work includes domain adaptation for document category specific HMMs such that we can adapt a trained model from one corpus to another corpus.

References

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

[2] T. Braun, F. Kuhr, and R. Möller, “Unsupervised text annotations,” in *Formal and Cognitive Reasoning - Workshop at the 40th Annual German Conference on AI (KI-2017)*, 2017.

[3] F. Kuhr, B. Witten, and R. Möller, “Corpus-driven annotation enrichment,” in *13th IEEE International Conference on Semantic Computing, ICSC 2019, Newport Beach, CA, USA, January 30 - February 1, 2019*, 2019, pp. 138–141.

[4] J. Yang, Y. Zhang, L. Li, and X. Li, “YEDDA: A lightweight collaborative text span annotation tool,” in *Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, System Demonstrations*, 2018, pp. 31–36.

[5] J. K. Kummerfeld, “Slate: A super-lightweight annotation tool for experts,” *arXiv preprint arXiv:1907.08236*, 2019.

[6] D. Collarana, M. Galkin, I. T. Ribón, M. Vidal, C. Lange, and S. Auer, “MINTE: semantically integrating RDF graphs,” in *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, WIMS 2017, Amantea, Italy, June 19-22, 2017*, 2017, pp. 22:1–22:11.

[7] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: a core of semantic knowledge,” in *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, 2007, pp. 697–706.

[8] K. Papantoniou, G. Tsatsaronis, and G. Paliouras, “KDTA: automated knowledge-driven text annotation,” in *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III*, 2010, pp. 611–614.

[9] X. Liao and Z. Zhao, “Unsupervised approaches for textual semantic annotation, a survey,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–45, 2019.

[10] T. Reuters, “Opencalais,” *Retrieved June*, vol. 16, 2008.

[11] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, “Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia,” *Semantic Web*, vol. 6, no. 2, 2015.

[12] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

[13] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, and W. Zhang, "From data fusion to knowledge fusion," *PVLDB*, vol. 7, no. 10, pp. 881–892, 2014.

[14] L. Getoor and C. P. Diehl, "Link mining: a survey," *SIGKDD Explorations*, vol. 7, no. 2, pp. 3–12, 2005.

[15] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

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

[17] H. B. Newcombe, J. M. Kennedy, S. Axford, and A. P. James, "Automatic linkage of vital records," *Science*, vol. 130, no. 3381, pp. 954–959, 1959.

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

[19] 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.

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

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

[22] 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.

[23] 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.

[24] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[26] 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.

[27] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 721–741, 1984.

[28] M. D. Hoffman, D. M. Blei, and F. R. Bach, "Online learning for latent dirichlet allocation," in *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, 2010, pp. 856–864.

[29] F. Kuhr, M. Bender, T. Braun, and R. Möller, "Maintaining Topic Models for Growing Corpora," in *In Proceedings of the 14th IEEE International Conference on Semantic Computing*. Springer, 2020.