To Extend or not to Extend? Enriching a Corpus with Complementary and Related Documents

Magnus Bender, Felix Kuhr
Institute of Information Systems, University of Lübeck,
Ratzeburger Allee 160, Lübeck, Germany
{*bender, kuhr*}*@ifis.uni-luebeck.de*
http://www.ifis.uni-luebeck.de

Tanya Braun
Computer Science Department, University of Münster,
Einsteinstr. 62, Münster, Germany
tanya.braun@uni-muenster.de
http://www.uni-muenster.de/Informatik

An agent in pursuit of a task may work with a corpus of documents with linked subjective content descriptions. Performing the task of document retrieval for a user or aiming to extend its own corpus, an agent so far relies on similarity measures to identify related documents. However, similarity may not be appropriate if looking for new information or different aspects of the same content. Therefore, this paper combines complementarity- and similarity-based identification of documents, specifically, contributing (i) a formal definition of complementarity using the available subjective content descriptions in the form of relational tuples as well as a taxonomy interrelating the concepts of the tuples, (ii) a technique for classifying complementary and related documents in one go, and (iii) a case study assessing the classification performance for complementary and related documents.

Keywords: Subjective content descriptions; corpus enrichment; complementarity; text mining.

1. Introduction

An agent in pursuit of a task may work with an individual collection of documents (corpus) as a reference library. We assume that the individual collection of documents represents a specific context in which the agent performs its task and documents are associated with location-specific subjective content descriptions (SCDs) making the content explicit by providing additional data in support of the agent's task. As part of its service, the agent may search for new documents to extend its corpus, e.g., to add new information or provide a well-rounded collection of documents given a user request for document retrieval. We refer to this internal task of an agent as corpus extension.

To decide a corpus extension, the agent has to determine if a document is related to a corpus. Relatedness can be captured by some measure of similarity, defined using words directly or representations derived from them such as topic-word

probability distributions, inferring abstract topics represented as distributions over a vocabulary, or SCD-word probability distributions, representing how often words appear around locations of SCDs. Kuhr et al. [1, 2] have worked with four document categories based on similarity using SCD-word probability distributions: (i) quasi copies, a.k.a. similar documents, (ii) extensions, (iii) revisions, and (iv) unrelated documents. However, classifying documents on similarity may lead to looking at documents that only contain more of the same, albeit possibly updated information.

To avoid being stuck in this bubble of similarity, we need to define a different measure of relatedness. Therefore, we focus on adding a fifth document category *complement*, which is hard to define given only words or numbers in distributions or vector representations. Complements may use a completely different vocabulary, which may render it as an unrelated document given similarity measures based on words. In terms of vector representations, one may think of a complement as a document having high values in certain dimensions where another one has low values. This consideration may also apply to completely unrelated documents, though, making it a not very effective measure. Therefore, we consider two problems, (i) formally defining complementary documents and distinguishing complementary documents from unrelated documents, and (ii) extending the document classification technique by Kuhr et al. [1, 2] with a fifth document category *complement*.

To get a handle on complementarity by way of a formal definition, we turn to SCDs, specifically, SCDs in the form of relational tuples such as subject-predicate-object (SPO) tuples together with a taxonomy that specifies a concept hierarchy for the constants occurring in SCDs. We hypothesize the following: Complementary documents have SCDs that contain different constants of the same concept in a taxonomy. Given this hypothesis, we can formally define complementary documents and specify a corresponding document classification problem. Given a definition of complementarity, we can solve the first problem of distinguishing unrelated and complementary documents by calculating a complementarity value between two documents based on their SCDs and how they interrelate given a taxonomy.

Facing the second problem of extending the document classification technique by Kuhr et al. [1, 2] with a category *complement*, we extend the SCD-word probability distribution with *complementary* SCDs originating from complementary documents. Thus, the combined SCD-word probability distribution contains related and corresponding complementary SCDs. For a new document to classify, the agent is then able to estimate most probable related and complementary SCDs using the combined SCD-word distribution. Altogether, the agent uses the estimated SCDs to classify a document in one of the five categories.

Specifically, the contributions of this paper are:

(i) a definition of the document classification problem for complements and a definition of complementarity for SCDs in the form of relational tuples and a definition of complementarity for documents based on complementary SCDs,

(ii) a first solution approach to the problem, which can be used for distinguishing unrelated and complementary documents,
(iii) an altogether document classification technique for complementary and related documents based on Kuhr et al., and
(iv) a case study on the performance of distinguishing unrelated and complementary documents as well as the overall document classification performance, comparing this article's approach against the method by Kuhr et al.

The remainder of this paper is structured as follows: We start with related work followed by a specification of notations and a recap of SCDs and document categories. Then, we specify complementarity based on SCDs and present a solution approach to identify complementary documents. Next, we present how to integrate complementary SCDs in the SCD-word distribution and how to classify documents using this distribution. Finally, we present a case study and end with a conclusion.

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 [3], Slate [4], MINTE [5], and YAGO [6]. For further annotation systems, please refer to [7]. Some annotation systems like OpenCalais [8] automatically attach data from a knowledge base (KB), e.g., DBpedia [9], to extractable named entitys (NEs) in the text. That is, the extractable NEs are matched to data in a KB to add data from the KB to the document. Adding data to documents might increase the performance of a document retrieval system.

In this paper, we investigate a different but related problem, namely estimating the complementarity of a new document with respect to the documents in a reference library of an agent. The complementarity of a document is based on NEs extractable from the text of the document and the NEs available in documents from the reference library. An agent can decide to extend a reference library with a new document complementary to documents in its library. In general, other automatic annotation systems ignore the context of a reference library and add data to documents already available in the reference library of an agent.

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 [10], 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) [11], comparing an estimated topic distribution of a new document with topic distributions of documents in a given corpus, or (iii) entity matching [12] using named-entity recognition (NER), comparing entities (and relations) retrieved from the new document with entities (and relations) from SCDs in the corpus. We aim at providing an approach to estimating

the complementarity of a new document using a given concept hierarchy and entities. The first two approaches have drawbacks regarding identifying complementary documents: Both are bag-of-words approaches, i.e., they ignore the order of words and extractable NE. Thus, we use elements from entity matching to link entities from a document to an external concept hierarchy.

Another class of related work deals with HMM-based classification. Classification and statistical learning using hidden Markov models (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 [13], character recognition [14], finance data prediction [15, 16], credit card fraud detection [17], and workflow mining [18]. Most systems learn an HMM by the Baum-Welch algorithm [19], which is a special case of the EM algorithm [20]. The goal of an HMM is estimating the most likely sequence of hidden states in a dynamic programming fashion by the Viterbi algorithm [21].

3. Preliminaries

This section specifies notations, defines SCD-word probability distributions, and recaps how an SCD-word probability distribution can be used to classify documents.

3.1. *Notation*

We define the following terms to formalize the setting of a corpus containing documents, where each document is associated with SCDs.

- A word w is a basic unit of discrete data from a vocabulary $\mathcal{V} = (w_1, \ldots, w_V)$, $V \in \mathbb{N}$.
- A document d is a sequence of words $(w_1^d, \ldots, w_N^d)$, $N \in \mathbb{N}$. Function $\#words(d)$ returns the total number of words in d, i.e., N.
- A corpus $\mathcal{D}$ refers to a set of documents $\{d_1, \ldots, d_D\}, D \in \mathbb{N}$, and $\mathcal{V}_\mathcal{D}$ to the corpus-specific vocabulary containing the words occurring in $\mathcal{D}$.
- An SCD t is a relational tuple of the form subject-predicate-object and t can be associated with a position ρ in a document d. We 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 d that t is associated with.
- For each located SCD $t_j \in g(d)$ exists a corresponding SCD window $win_{d,\rho}$ referring to a sequence of words in d. In our case $win_{d,\rho}$ belongs to the words forming the sentence number ρ in in d.
- 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 $g(\mathcal{D}) = \bigcup_{d \in \mathcal{D}} g(d)$.

- Each word $w^d \in win_{d,\rho}$ is associated with an influence value $I(w^d, win_{d,\rho})$ representing the distance 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 $I(w^d, win_{d,\rho})$ is. Generally, the function to estimate the influence value of a word depends on the specific task of an agent.

3.2. *SCD-Word Probability Distributions*

We define an additional representation for each SCD by taking a vector of length V, $V = |\mathcal{V}_\mathcal{D}|$, where each vector entry refers to a word in the vocabulary $\mathcal{V}_\mathcal{D}$ of corpus $\mathcal{D}$. The vector entry itself is a probability value describing how likely it is that a word occurs in an SCD window surrounding the position associated with the SCD, yielding an SCD-word probability distribution for each SCD associated with documents in $\mathcal{D}$. In other words, 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 exclusively associated with a single document in a corpus and might occur more than once.

A SCD-word probability distribution can be generated by counting the occurrences of words around SCDs weighted with their influence value [1]. Equation (1) shows the SCD-word probability distribution as an $m \times V$ matrix $\delta(\mathcal{D})$, with the SCD-word probability distribution vectors forming the rows of the matrix:

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

Kuhr et al. [1] generate the matrix for a corpus of related documents, thus, we call their SCD-word probability distribution related SCD (rSCD) matrix $\delta_r(\mathcal{D}_r)$.

Faced with a document having no SCDs, an agent may want to enrich the document with SCDs from the corpus based on the SCD-word probability distribution. To this end, the agent divides the new document into M windows and generates a word frequency vector from the words in each window. The agent compares the word frequency vector of each window with the word frequency vector of each SCD associated with documents in the corpus, i.e., the rows in the matrix. The SCD where the word frequency vector has the smallest distance (highest cosine similarity) to the word frequency vector of the window is associated with the window. We refer to this associated SCD as the most probably suited subjective content description (MPSCD). Next, we recap how MPSCDs based on an rSCD matrix can be used to classify documents.

3.3. *Corpus Extension using Similarity*

Using the MPSCD similarity values, Kuhr et al. [1, 2] present a method with which an agent can classify a new document d' by one of the following four categories:

- *Quasi copy*: Document d' is classified as *sim* if the values in the MPSCD similarity sequence are mostly high and contain only few entries with slightly lower values.
- *Extension*: Document d' is classified as *ext*, representing an extension of another document $d \in \mathcal{D}$, if d' is generated by *appending* a document d, i.e., d' represents an updated version of d.
- *Revision*: Document d' is classified as rev, representing a revision of another document $d \in \mathcal{D}$ generated by *replacing* or *removing* parts of d.
- *Unrelated document*: Document d' is classified as *unrel* if the values in the MPSCD similarity sequence of d' are mostly low.

In the next section, we define complementarity of SCDs and documents leading to a new category *complement* of documents. Then, we describe an approach classifying documents of the new category *complement* based on the definitions.

4. Identifying Complementary Documents

This section presents an approach for identifying documents containing complementary content with respect to the content of documents in a given corpus. We use the task of corpus extension as the application scenario for complementary documents. However, the given definitions and algorithms can be tweaked with minimal effort for other tasks such as document retrieval. First, we define a binary document classification problem, i.e., if a document is complementary. Second, we provide a definition of complementary documents based on SCD complementarity values to solve the problem. Third, we present an approach to corpus extension with complementary documents by identifying a new document d' as a complement using the previously defined notions.

4.1. *Document Classification Problem: Complement*

Given an unknown document d' and a corpus $\mathcal{D}$, an agent might be interested in whether d' is a *complement* to documents in $\mathcal{D}$. Formally, we ask whether d' is a complement to d ($Complement = true$) or not ($Complement = false$), making the document classification problem a binary classification problem

$$\underset{v \in \{true, false\}}{\arg\max} \; P(Complement = v \mid d', \mathcal{D}). \tag{2}$$

Since it is non-trivial to get the necessary probability distributions, we solve the problem of Eq. (2) by looking at SCDs, defining complementarity in terms of SCDs and a complement by using the notion of complementary SCDs. Based on these definitions, we specify a solution approach for corpus extension.

4.2. *Complementary Documents*

To classify a document as a complement, i.e., providing complementary content, with respect to the documents in the corpus of an agent, we need a formal definition of complementarity, for which we use the SCDs that are available in an SPO format and a taxonomy for interrelating entities occurring in them. As such, we transform the problem given in Eq. (2) by defining a complement as a document with a complementarity value that exceeds a certain threshold. Focussing on SCDs in the SPO format also has the upside that we can automatically extract relational structures using available NE extraction methods such as OpenIE [22] to generate SCDs for documents. We can even use a lexical database of semantic relations and use hierarchies to interrelate those entities.

Before defining complementary SCDs and documents, let us consider an example of a new document containing complementary content to the content of documents in a corpus. We pick up the example again in the course of this article.

Example 1. Assume that an agent is working with an individual collection of documents in corpus $\mathcal{D}$. The documents contain text about competitions at the Olympic Games 2021 in Tokyo. Thus, vocabulary $\mathcal{V}_{\mathcal{D}}$ is mainly characterized by words in the context of sports. The vocabulary of a new document d' giving a description about the occurrence of infection of `SARS-CoV-2` in Tokyo is different from $\mathcal{V}_{\mathcal{D}}$. In the context of similarity, d' would probably be classified as unrelated since the vocabularies $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{d'}$ might be very different. However, the content of d' might be complementary to the content of some documents in $\mathcal{D}$ and thus, might support an agent to interpret content from documents in $\mathcal{D}$ more suitably.

Definition 1 (Complementary SCDs). Given two documents d, d' and a taxonomy ξ, an SCD $t_i \in g(d')$ is complementary to an SCD $t_j \in g(d)$ if the entities in t_i and t_j are different but the entities are instances of the same concept or the predicates between the entities share a common meaning in ξ. Formally, the following seven types of complementarity between SCDs t_i and t_j exist ($\uparrow$ refers to the concept in ξ that an entity belongs to):

(1) `s`-complementary: $t_i = (s^{\uparrow}, p_i, o_i)$, $t_j = (s^{\uparrow}, p_j, o_j)$,
(2) `p`-complementary: $t_i = (s_i, p^{\uparrow}, o_i)$, $t_j = (s_j, p^{\uparrow}, o_j)$,
(3) `o`-complementary: $t_i = (s_i, p_i, o^{\uparrow})$, $t_j = (s_j, p_j, o^{\uparrow})$,
(4) `sp`-complementary: $t_i = (s^{\uparrow}, p^{\uparrow}, o_i)$, $t_j = (s^{\uparrow}, p^{\uparrow}, o_j)$,
(5) `so`-complementary: $t_i = (s^{\uparrow}, p_i, o^{\uparrow})$, $t_j = (s^{\uparrow}, p_j, o^{\uparrow})$,
(6) `op`-complementary: $t_i = (s_i, p^{\uparrow}, o^{\uparrow})$, $t_j = (s_j, p^{\uparrow}, o^{\uparrow})$, and
(7) `spo`-complementary: $t_i = (s^{\uparrow}, p^{\uparrow}, o^{\uparrow})$, $t_j = (s^{\uparrow}, p^{\uparrow}, o^{\uparrow})$.

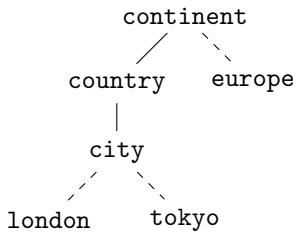
Let $\mathcal{X}$ refer to the set of the different complementarity types $\{\mathtt{s}, \mathtt{p}, \mathtt{o}, \mathtt{sp}, \mathtt{so}, \mathtt{op}, \mathtt{spo}\}$. An indicator function $\mathfrak{C}_x(t_i, t_j)$, $x \in \mathcal{X}$, returns 1 if t_i and t_j fulfil the conditions mentioned above for x-complementarity and otherwise 0, including when t_i or t_j is not in SPO format.

Generally, it might be possible to adapt the return value of the indicator function to include uncertainty by returning a value from $[0, 1]$. Next, we give an example on complementary SCDs.

Example 2 (Complementary SCDs). Assume that document d is in the agent's corpus and the agent is faced with a new document d'. Additionally, both documents are associated with SCDs yielding $g(d) = \{t_2, t_4\}$ and $g(d') = \{t_1, t_3\}$ where:

- $t_1 = (Olympic\ Games\ 2021, in, Tokyo)$,
- $t_2 = (SARS\text{-}CoV\text{-}2, spreading\ in, Tokyo)$,
- $t_3 = (UEFA\ Euro\ 2020, in, Europe)$, and
- $t_4 = (Covid\text{-}19, spreading\ in, London)$

Given the following taxonomy, where solid lines represent the hierarchy between classes and dashed lines represent instances of classes,



the indicator function $\mathfrak{C}_o(t_i, t_j)$ returns 1 for $i = 1$ and $j = 4$ since both `london` and `tokyo` are instances of class `city`. Additionally, the indicator function returns 1 for $i = 3$ and $j = 4$ since `london` is a `city` and `city` is a subclass of `continent`, to which `europe` belongs, too. Thus, t_1 and t_4 as well as t_3 and t_4 are `o`-complementary.

The different types of complementarity form a lattice as depicted in Fig. 1 with the first three types composing the lowest level, the next three types following on the next higher level, and the `spo`-type constituting the top entry. Up the lattice, the SCDs share more and more entities of the same concept, with the top entry requiring that the three positions are filled with different instances of the same concept, i.e., what falls under complementarity of higher levels also falls under complementarity of lower levels. This is different to Def. 1 of [23], which requires all entities to share the same concept or be *identical*. The new Def. 1 requires the entities to share the same concept or be *different*, thereby, further moving away from similarity to difference. A complementary SCD is now understood as a different SCD only sharing one or multiple concepts and not an identical SCD allowed to share one or multiple concepts. Thus, the deviation of related and complementary SCDs will be much larger and the word vectors in the windows associated with the two types of SCDs will be more distinct. Next, we define complementary documents based on Def. 1.

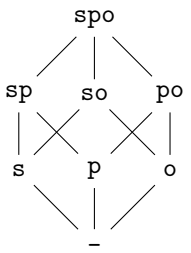


Fig. 1. The complementarity types of Def. 1 in a lattice.

Definition 2 (Complement). The complementarity value $\mathfrak{c}(d', d)$ between documents d' and d is given by

$$\mathfrak{c}(d', d) = \sum_{t_i \in g(d')} \sum_{t_j \in g(d)} \sum_{x \in \mathcal{X}} w_x \mathfrak{C}_x(t_i, t_j), \tag{3}$$

with $w_x \in [0, 1]$ a weight assigned to each complementarity type and $\sum_{x \in \mathcal{X}} w_x = 1$. Given a threshold θ_d, d' is complementary to d and thus called a *complement* if

$$\mathfrak{c}(d', d) > \theta_d. \tag{4}$$

Given the complementarity lattice, non-zero weights are only reasonable for types that do not subsume another. E.g., given spo-complementarity, w_x should be set to zero for all other complementarity type, i.e., $\forall x \in \mathcal{X}, x \neq \texttt{spo}$, as these types x would subsume spo. For the lowest level, $w_x = 0$ for all $x \in \{\texttt{sp},\texttt{so},\texttt{op},\texttt{spo}\}$ while $w_\texttt{s}$, $w_\texttt{p}$, and $w_\texttt{o}$ can be chosen freely as long as they add up to 1. Another possibility would be to have non-zero weights adding up to 1 for, e.g., sp and o, as they cover different positions in the triples and lie on different paths in the grid, with the remaining x set to 0. The threshold θ_d depends on a given corpus and may reflect the agent's need for new documents. With a need for more documents, an agent may choose a low threshold in combination with the broadest senses of complementarity, s, p and o. Aiming for adding only a few documents in the more immediate context, a high threshold and spo-complementarity might be a fitting choice. In Example 2, $\mathfrak{c}(d', d) = 3$ with $w_\texttt{o} = 1$ ($\mathfrak{C}_\texttt{o}(t_2, t_1) = 0$). In all other cases, $\mathfrak{c}(d', d) = 0$ as $\forall x \neq \texttt{o} : \mathfrak{C}_x(t_i, t_j) = 0$ with the present taxonomy.

We use the decision criterion in Eq. (4) to solve the document classification problem of Eq. (2). Within the framework of Kuhr et al.'s document classification problem, we could focus computing Eq. (4) for those documents that are otherwise classified as unrelated, making a distinction between complement and unrelated How well this definition works for distinguishing unrelated and complementary documents, we showcase during the case study of Section 6. But before that, we present how to use the definitions of complementarity for the task of corpus extension and briefly discuss what else can be done with the setting available.

Algorithm 1 Corpus Extension with Complements

```
function extendComplement(𝒟, d′, θ_𝒟, {w_x}_{x∈𝒳})
    Input: Corpus 𝒟, new document d′, threshold θ_𝒟, weights {w_x}_{x∈𝒳}
    Output: true (complement/extend) or false (no complement/extend not)
    if g(d′) = ∅ then
        Add SCDs to d′ using OpenIE
    c ← 0
    for each t_i ∈ g(d′) do
        for each d ∈ 𝒟 do
            for each t_j ∈ g(d) do
                for each x ∈ 𝒳 do
                    c ← c + w_x 𝔠_x(t_i, t_j)
    if c > θ_𝒟 then
        return true
    return false
```

4.3. *Corpus Extension with Complements*

Corpus extension as a task so far has used similarity values, specifically the sequence of MPSCD similarity values over a document, to classify an unknown document as either of the document types of *sim*, *ext*, *rev*, and *unrel*, and then decide its inclusion based on this outcome. Similar and unrelated documents were ignored whereas extensions and revisions were added or exchanged with the originals. An agent performing corpus extension with complementary documents has to answer the same question about possibly including an unknown document. However, now the agents aims to extend its corpus with complements. To perform the task, the agent applies the definitions above for reaching a decision.

Algorithm 1 shows an outline of the workflow the agent follows when presented with an unknown document d' for possible inclusion into its corpus $\mathcal{D}$ on the condition that d' is a complement in $\mathcal{D}$. The algorithm uses a corpus-specific threshold $\theta_{\mathcal{D}}$, which fulfils the same role as the threshold θ_d in Eq. (4) but factors in that it applies to the whole corpus and not a single document. The first if-condition asks whether d' already contains SCDs. If not, the agent uses OpenIE to extract SPO tuples from the text of d'. Then follows a for-loop that accumulates the complementarity values for each SCD t_i associated with d' over all documents in $\mathcal{D}$. Afterwards, the agent tests the accumulated value against $\theta_{\mathcal{D}}$ to return *true* if it considers d' a complement based on Def. 2, and $false$ otherwise.

4.4. *Discussion*

The following paragraphs discuss complements as part of the general classification problem, for the task of document retrieval, and for augmenting user output by returning positions of interest.

Complements as a Document Type While we provide a more general approach in the upcoming section, a direct way to introduce complements as category *compl* into the corpus extension by Kuhr et al. [1, 2] is the following: The classification problem in [1, 2] is defined given a sequence of MPSCD similarity values $\mathcal{W}$ computed by a version of Alg. 4 for an unknown document d' and a corpus $\mathcal{D}$:

$$\arg\max_{y \in \mathcal{Y}} P(Type = y \mid \mathcal{W}), \tag{5}$$

with $\mathcal{Y} = \{sim, ext, rev, unrel\}$. Generalizing and merging Eqs. (2) and (5), we could formulate the classification problem as follows:

$$\arg\max_{y \in \mathcal{Y}} P(Type = y \mid d', \mathcal{D}), \tag{6}$$

with $\mathcal{Y} = \{sim, ext, rev, unrel, compl\}$. In Eq. (6), an unknown document and the corpus are given. A reasonable workflow to classify an unknown document would then be to use the document type detection algorithm in [1, 2] and then apply Alg. 1 to the unknown document if the previous classification returns *unrel*.

Document Retrieval For document retrieval in the context of complementarity, i.e., complement retrieval, a user could provide a document d' for which they want k complementary documents returned from the corpus $\mathcal{D}$ available to the agent. The agent would then calculate complementarity values for d' compared to each document $d \in \mathcal{D}$, i.e., $\mathfrak{c}(d', d)$ following Def. 2, and return the top-k documents, i.e., those k documents with the highest complementarity values. In contrast to Alg. 1, the agent would not accumulate the complementarity values but rather store the current top-k documents with their complementarity value and test whether the next document d has a higher value than the lowest value currently stored and replace that document if true.

Augmenting Enrichment: Positions of Interest In general, it is difficult to understand the reason a new document is classified as a complementary document by looking at the content of the document. The only thing we know for a document being classified as complementary is that some entities from a new document share a class with entities from documents in the corpus. Thus, one might highlight complementary SCDs s.t. it is possible to identify the positions in a text that are relevant for Alg. 1 classifying a document as a complementary document. We denote those positions as *positions of interest*.

With the definition of complementarity in place and a solution approach specified with Alg. 1, we are able to detect complementary documents. However, the classification process is not straightforward, as first the documents are classified as one of $\{sim, ext, rev, unrel\}$. Then, if a document is classified as *unrel*, a second classification between $\{unrel, compl\}$ is performed. In the next section, we describe how to integrate complementarity in the SCD-word distribution matrix, allowing for directly classifying documents as as one of $\{sim, ext, rev, unrel, compl\}$.

5. Document Classification with Complementarity and Similarity

The SCD-word distribution matrix used by Kuhr et al. [1, 2], called rSCD matrix, only contains related SCDs and the authors classify a new document d' by the four document types $\{sim, ext, rev, unrel\}$ using similarity. To support complementarity, we propose a combined SCD (cSCD) matrix, containing related and complementary SCDs in one matrix. The cSCD matrix allows an agent to classify a new document d' by five document types $\{sim, ext, rev, unrel, compl\}$ in one classification process. First, we present an algorithm to build a cSCD matrix, followed by a filtering technique removing noisy SCDs from a cSCD matrix using the definition of complementarity, and finally, a straightforward classification process.

5.1. *Combined SCD Matrix*

The cSCD matrix combines two corpora: $\mathcal{D}_r$ contains the related documents from the agent's corpus, the same corpus Kuhr et al. train their matrix on. Additionally, the cSCD matrix is trained on $\mathcal{D}_c$ containing complementary documents to the agent's corpus. Corpus $\mathcal{D}_c$ can be either formed by using Alg. 1 or by using expert's knowledge, manually forming a corpus of complementary documents. Analogously to the general notations, each corpus has a set of SCDs $t_j^r \in g(\mathcal{D}_r)$ and $t_j^c \in g(\mathcal{D}_c)$. The vocabularies of both corpora are joined, i.e., $\mathcal{V} = \mathcal{V}_{\mathcal{D}_r} \cup \mathcal{V}_{\mathcal{D}_c}$ and $V = |\mathcal{V}|$. The cSCD matrix $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$ is a more specific form of $\delta(\mathcal{D})$ shown in Eq. (1):

$$\delta_c(\mathcal{D}_r, \mathcal{D}_c) = \begin{array}{c} \\ t_1^r \\ \vdots \\ t_{m_r}^r \\ t_1^c \\ \vdots \\ t_{m_c}^c \end{array} \begin{array}{c} \begin{array}{ccccc} w_1 & w_2 & w_3 & \cdots & w_V \end{array} \\ \begin{pmatrix} v_{1,1} & v_{1,2} & v_{1,3} & \cdots & v_{1,V} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{m_r,1} & v_{m_r,2} & v_{m_r,3} & \cdots & v_{m_r,V} \\ v_{m_r+1,1} & v_{m_r+1,2} & v_{m_r+1,3} & \cdots & v_{m_r+1,V} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{m_r+m_c,1} & v_{m_r+m_c,2} & v_{m_r+m_c,3} & \cdots & v_{m_r+m_c,V} \end{pmatrix} \end{array} \tag{7}$$

The first m_r rows belong to the SCDs $g(\mathcal{D}_r)$ and the last m_c rows belong to $g(\mathcal{D}_c)$. Again, each row in the matrix forms an SCD-word probability distribution vector. The vector entry itself is a probability value describing how likely it is that a word occurs in an SCD window surrounding the position associated with the SCD, yielding an SCD-word probability distribution for each SCD associated with documents in $\mathcal{D}_r$ or $\mathcal{D}_c$, respectively.

Furthermore, associated with a cSCD matrix is a set of complementarity relations between the SCDs of the matrix:

$$\mathcal{C}_x = \{(\{t_i^r, t_j^c\}, \mathfrak{C}_x(t_i^r, t_j^c)) \mid t_i^r \in g(\mathcal{D}_r), t_j^c \in g(\mathcal{D}_r)\}$$

where $x \in \mathcal{X}$ refers to a complementarity type and $\mathfrak{C}_x$ returns a continuous value from $[0, 1]$. Note that the complementarity value between two SCDs is symmetric,

Algorithm 2 Building cSCD matrix $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$

function BUILDCOMBINEDMATRIX($\mathcal{D}_r$, $\mathcal{D}_c$, x)
  Input: Corpora $\mathcal{D}_r$, $\mathcal{D}_c$, complementarity type x
  Output: cSCD matrix $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$, complementarity relations $\mathcal{C}_x$
  Initialize an $(m_r + m_c) \times V$ matrix $\delta(\mathcal{D})$ with zeros
  for each $d \in \mathcal{D}_r \cup \mathcal{D}_c$ **do** ▷ Form word distributions
    for each $t \in g(d)$ **do**
      for sentence $win_{d,\rho}$ of t **do**
        for each word $w \in win_{d,\rho}$ **do**
          $\delta_c(\mathcal{D}_r, \mathcal{D}_c)[t] \mathrel{+}= I(w, win_{d,\rho})$
  Normalize $\delta_c(\mathcal{D}_r, \mathcal{D}_c)[t]$
  Initialize $\mathcal{C}_x \leftarrow \emptyset$
  for each $t^r \in g(\mathcal{D}_r)$ **do** ▷ Extract complementarity relations
    for each $t^c \in g(\mathcal{D}_c)$ **do**
      $\mathcal{C}_x \leftarrow \mathcal{C}_x \cup \{(\{t^r, t^c\}, \mathfrak{C}_x(t^c, t^r))\}$ ▷ Indicator function from Def. 1
  return $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$, $\mathcal{C}_x$

i.e., $\mathfrak{C}_x(t_i, t_j) = \mathfrak{C}_x(t_j, t_i)$ for any SCDs t_i, t_j. The complementarity set $\mathcal{C}_x$ represents for each SCD the corresponding complementary SCDs together with the value of complementarity. Thus, given an SCD linked to a document, it is possible to retrieve the complementary SCDs and linked complementary documents.

5.1.1. *Building Combined SCD Matrices*

Algorithm 2 generates a cSCD matrix and the associated set $\mathcal{C}$ based on two corpora $\mathcal{D}_r$ and $\mathcal{D}_c$. The algorithm iterates over all SCDs of both corpora and updates for each sentence as window the word distribution vector based on the influence value I. Afterwards, the relations between the SCDs in the matrix are calculated applying Def. 1 and the complementarity values are stored in $\mathcal{C}$.

5.1.2. *Filtering Combined SCD Matrices*

A cSCD matrix formed by Alg. 2 contains a vector (row) for each SCD from both corpora. However, complementary documents also contain some false-complementary SCDs, i.e., SCDs for general sentences which can occur in documents of any context. This is similar to noisy data because false-complementary SCDs are considered as complementary when using the cSCD matrix, even though they are not. Depending on the use case, false-complementary SCDs might provide useful data but in our scenario they add noise to the results. Therefore, we introduce the filtered cSCD (cSCD$^{\mathrm{f}}$) matrix, in which the false-complementary SCDs are removed from the matrix. In the case study, we compare the performance of the cSCD and cSCD$^{\mathrm{f}}$ matrix for classification.

Algorithm 3 Filter cSCD matrix to get cSCD$^{\text{f}}$ matrix

function FILTERCOMBINEDMATRIX($\delta_c(\mathcal{D}_r, \mathcal{D}_c)$, $\mathcal{C}_x$, θ_δ)
  Input: cSCD matrix $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$, complementarity relations $\mathcal{C}_x$, threshold θ_δ
  Output: cSCD$^{\text{f}}$ matrix $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$
  for each $t^c \in g(\mathcal{D}_c)$ **do**
    $best \leftarrow 0$
    for each $t^r \in g(\mathcal{D}_r)$ **do**
      $value \leftarrow \mathcal{C}_x(t^c, t^r)$ ▷ Retrieve complementarity value from set $\mathcal{C}_x$
      $best \leftarrow \max\{value, best\}$
    if $best < \theta_\delta$ **then**
      Delete row $\delta_c(\mathcal{D}_r, \mathcal{D}_c)[t^c]$
  return $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$

During filtering, all SCDs in $g(\mathcal{D}_r)$ are kept. SCDs in $g(\mathcal{D}_c)$, which are not complementary by our definition of complementarity, are removed. We use a threshold θ_δ to decide if an SCD is complementary, which depends on the corpora, complementarity type, and cSCD matrix used. Algorithm 3 iterates over all SCDs $t^c \in g(\mathcal{D}_c)$ and extracts the highest complementarity value of each t^c to any SCD $t^r \in g(\mathcal{D}_r)$. If the highest value is smaller than θ_δ, the SCD t^c is considered to be false-complementary and the SCD-word distribution $\delta_c(\mathcal{D}_r, \mathcal{D}_c)[t^c]$ is removed in the cSCD$^{\text{f}}$ matrix.

To decide a document extension, an agent faced with an unknown document d' has to identify the document type $y \in \mathcal{Y} = \{sim, ext, rev, unrel, compl\}$ of d', with $\mathcal{Y}$ now containing *compl* as another type compared to previous settings. Therefore, we have to adapt existing procedures to this change, which also includes the switch from rSCD to cSCD or cSCD$^{\text{f}}$ matrices. As the existing procedure are based on MPSCDs and their similarity values over the text of d', we first consider how to estimate MPSCDs for a d' given a cSCD or cSCD$^{\text{f}}$ matrix of a corpus $\mathcal{D}_r \cup \mathcal{D}_c$.

5.2. *Estimating Most Probably Suited SCDs in cSCD Matrices*

To estimate MPSCDs, the agent generates a word frequency vector from the words of each sentence of the new document. The agent compares the word frequency vector of each sentence with the word frequency vector of each SCD associated with documents in the corpus. The SCD where the word frequency vector has the smallest distance (highest cosine similarity) to the word frequency vector of the sentence is associated with the sentence. We refer to this associated SCD as the MPSCD. An MPSCD may originate from the related or complementary corpus. Thus, a new document associated with mostly complementary MPSCDs might by a complementary document, while a new document associated with mostly related MPSCDs is more likely a similar or revised document.

Algorithm 4 outlines the procedure of estimating MPSCDs, which not only returns the MPSCDs but also the cosine similarity values of each MPSCD and hence

Algorithm 4 Estimating MPSCDs using cSCD and cSCD$^{\text{f}}$ matrices

function ESTIMATEMPSCD(d', $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$)
  Input: Document d', cSCD or cSCD$^{\text{f}}$ matrix $\delta_c(\mathcal{D}_r, \mathcal{D}_c)$
  Output: SCDs $g(d')$ with similarity values $\mathcal{W}$
  $\mathcal{W} \leftarrow \emptyset$
  for each sentence $\rho \in d'$ **do**
    $\delta(win_{d',\rho}) \leftarrow$ new zero-vector of length V
    for each word $w \in win_{d',\rho}$ **do**
      $\delta(win_{d',\rho})[w]$ += $I(w, win_{d',\rho})$
    $t \leftarrow \underset{t_i \in g(\mathcal{D}_r) \cup g(\mathcal{D}_c)}{\arg\max} \frac{\delta(\mathcal{D})[t_i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[t_i]| \cdot |\delta(win_{d',\rho})|}$ in $win_{d,\rho}$
    $s \leftarrow \underset{t_i \in g(\mathcal{D}_r) \cup g(\mathcal{D}_c)}{\max} \frac{\delta(\mathcal{D})[t_i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[t_i]| \cdot |\delta(win_{d',\rho})|}$ ▷ Cosine similarity
    if $t \in g(\mathcal{D}_c)$ **then** ▷ Negative value for complementary SCDs
      $s \leftarrow s \cdot -1$
    $g(d') \leftarrow g(d') \cup \{(t, \rho)\}$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(sim, \rho)\}$
  return $g(d')$, $\mathcal{W}$

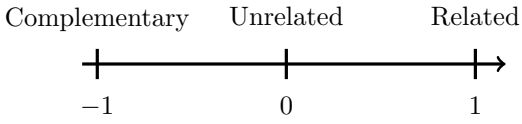


Fig. 2. Range of the MPSCD similarity values along with their interpretations.

of each sentence in d'. We call the sequence of cosine similarity values for a document MPSCD similarity sequence or in short MPSCD similarities. Each sentence is associated with a MPSCD similarity value.

During the estimation of MPSCDs, we map the similarity values of related SCDs to the interval $[0, 1]$ as before, with 0 representing *unrelated* SCDs. In contrast, complementarity is represented by the interval $[-1, 0]$ left of 0, with 0 representing *not complementary*, i.e., unrelated in terms of complementary SCDs, again. Thus, we overall get the interval $[-1, 1]$ for the MPSCD similarity values. The complementarity of sentences is now expressed by numbers in $[-1, 1]$, where -1 stands for complementary and 1 for related, with 0 the point of intersection for *unrelated* SCDs. Fig. 2 illustrates the interval of the MPSCD similarity values and Alg. 4 can be imagined as mapping each sentence to the interval.

5.3. *Classifying Documents*

Using the previously described techniques, a cSCD and cSCD$^{\text{f}}$ matrix can be trained and a sequence of MPSCDs similarities can be estimated for a new document.

Similarly to Kuhr et al., we apply an ensemble of HMMs to analyze the MPSCD similarities.

The MPSCD similarities contain for each sentence a similarity value, which aligns each sentence between complementary and related whereas unrelated is in the middle. Given this sequence describing a new document, we need to classify the document by five document types. For each type, we assume the following behaviour of the sequences:

unrel An unrelated document results in a sequence with mostly small values around 0. The word frequency vectors of the sentences neither match the vectors of the related nor the complementary SCDs in the matrix.

compl A complementary document results in a sequence with mostly high negative values close to -1. The word vectors of the complementary SCDs in the matrix are more similar to the vectors created on the sentences.

sim While a complementary document results in negative values, a similar document results in high positive values close to 1.

ext An extended document shows two sectors: The sequence starts with high positive values and ends with smaller or even negative values, if extended with complementary content.

rev In a revised document sentences have been replaced by complementary or unrelated content while the remaining sentences remain related. Thus, the sequence contains high positive, high negative, and even small values.

According to our assumptions about the sequences, we define a suitable HMM.

Definition 3 (Hidden Markov model). An hidden Markov model λ for classifying documents is a tuple $(\Omega, \Delta, A, B, \pi)$ consisting of

- (hidden) states $\Omega = \{s_1, ..., s_n\}$, where $n = 3$, with state s_1 representing complementary, s_2 unrelated, and s_3 related sentences,
- an observation alphabet $\Delta = \{o_{-m}, \ldots, o_0 \ldots, o_m\}$, where each o_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 $a_{i,j}, i, j \in \{1, 2, 3\}$ between the three states $s_1, s_2, s_3 \in \Omega$, 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:

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

Algorithm 5 Classification using an ensemble of HMMs and MPSCD similarities

```
1: function classifyDocument(d′, δ_c(D_r, D_c), H)
2:     Input: Document d′, cSCD or cSCD^f matrix δ_c(D_r, D_c), HMMs H
3:     Output: Type y ∈ Y of document d′
4:     y ← initialize, p ← 0
5:     W ← estimateMPSCD(d′, δ_c(D_r, D_c))
6:     O ← discretize(W)
7:     for each HMM λ_y ∈ H do
8:         S ← Viterbi(λ, O)
9:         if prob(S) > p then
10:            p ← prob(S)
11:            y ← λ_y
12:    return y
```

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

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

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

We compose an HMM of three hidden states because we assume each sentence represented may be related, unrelated, or complementary. The discrete observation alphabet Δ requires discretizing the sequences of MPSCD similarities. A discretization function $f : [-1, 1] \mapsto \Delta$ maps each MPSCD similarity value s to one of the symbols in Δ based on m thresholds $th_1, ..., th_m$:

$$f(s) = \begin{cases} o_{-m} & -1 \leq x < -th_m \\ \vdots & \\ o_0 & -th_1 \leq x < th_1 \\ \vdots & \\ o_m & th_m \leq x \leq 1 \end{cases} \tag{8}$$

In general, the transition probability matrix A and the emission probability matrix B are unknown and have to be learned, e.g., using the Baum-Welch algorithm [24]. Using a set of documents with known document type $y \in \mathcal{Y} = \{sim, ext, rev, unrel, compl\}$ we calculate the MPSCD similarities, discretize them, and train an HMM for each document type. The resulting ensemble of five HMMs

$$\mathcal{H} = \{\lambda_y | y \in \mathcal{Y}\}$$

is used by Alg. 5 to classify a new document.

Algorithm 5 estimates and discretizes the MPSCD similarities for a new document and runs the Viterbi algorithm [21] for each HMM in $\mathcal{H}$. The Viterbi algorithm

calculates the most probable sequence of hidden states on each HMM for the given sequence of observation symbols. Also, the probability for the sequence of hidden states is calculated. Finally, a document is classified as the document type for which the document type's HMM yielded the highest probability.

In summary, this section presents an improved version of the document classification by Kuhr et al. [2] to also recognize complementarity. In the next section, we provide a case study comparing the classification performance of documents using the rSCD, cSCD, and cSCD$^{\mathrm{f}}$ matrix.

6. Case Study

In this section, we present a case study illustrating the potential of the definition of complementarity and the classification approach using cSCD and cSCD$^{\mathrm{f}}$ matrices. We demonstrate that document classification using only an rSCD matrix is not able to detect complementary documents well. We show that the cSCD and cSCD$^{\mathrm{f}}$ matrices perform significantly better on the problem. Before we look at the results, we describe the corpus and workflow used in the case study.

6.1. *Corpus*

In this case study, we use articles from the English Wikipedia as documents in a corpus. All documents in the corpus contain text about car manufacturers[a]. Thus, documents about car manufacturers are related documents. We manually create document extensions by concatenating related and unrelated documents. To create a revised version of a document, we replace 40% of the sentences in related documents with sentences from unrelated documents. The class of unrelated documents contains the following 16 Wikipedia articles: *Apple Inc.*, *Apple*, *IPhone*, *Microsoft Windows*, *Google*, *Donald Trump*, *Atlantic Ocean*, *Angela Merkel*, *Baltic Sea*, *SpaceX*, *Lawyer*, *Titanic*, *Management*, *President (government title)*, *Mountain*, and *Snow*. Wikipedia articles about the cities where each of the car manufacturers' headquarters are located act as complementary documents. For example, the document *Toyota City, Aichi, Japan* is complementary to *Toyota Motor*.

Generally, the context of the corpus we are interested in can be described by *cars and their manufacturers*. Unrelated documents like *Apple Inc.* do not represent the manufacturing of cars and a profession like *Lawyer* neither represents cars nor manufacturing. We argue that complementary documents used in the evaluation fulfill our definition of complements, as the production of cars influences the city where the manufacturer is located, e.g., employees working at the manufacturer will reside in the city, the manufacturer pays taxes, and geographical conditions or historical circumstances of the city may originate from the manufacturer. However, some of the unrelated documents might be also a bit complementary, e.g., *Apple Inc.*

[a] `https://w.wiki/4FUS`

contains a short paragraph about an autonomous car. In contrast, the fruit *Apple* contains no content about cars or manufacturers. In this manner, it is important to notice that our definition of complementarity is universal and is not dependent or trained on a specific corpus. However, the cSCD and cSCD$^{\mathrm{f}}$ matrices are trained to fit the selected corpus.

6.2. *Workflow and Implementation*

All algorithms are implemented using Python. We use OpenIE [22] to extract SCDs in the SPO format from each window over the word sequences from the articles. Additionally, we use the WordNet [25] interface, provided by the Natural Language Toolkit[b], to detect if entities share the same concept or a common meaning in the sense of Def. 1.

Our implementation is optimized for speed and runs on multiple processor cores. It uses the libraries Gensim[c], NumPy[d], SciPy[e] and Pomegranate[f]. We run all experiments in a Docker container on a machine featuring 8 Intel 6248 cores at 2.50GHz (up to 3.90GHz) and 16GB RAM.

Before forming the SCD matrices using Alg. 2, we preprocess all documents by (i) removing punctuation, (ii) lowercasing all characters, (iii) stemming words, (iv) tokenizing the result, and (v) eliminating tokens from a stop-word list containing 179 words. OpenIE and WordNet's morphological processing tool Morphy use their own default preprocessing.

Each technique to test has its own workflow: (i) *similarity-based* serves as baseline and equals Kuhr et al. [2], (ii) *complementarity-based* directly uses the definition of complementary documents, and (iii) *document classification* describes how we apply the cSCD and cSCD$^{\mathrm{f}}$ matrix.

Similarity-based Training and using a similarity-based rSCD matrix works similar to Algs. 2, 4 and 5 with three differences, (i) no set $\mathcal{C}$ is generated, (ii) the set $\mathcal{D}_c$ is empty, and (iii) the HMMs have only two states (related and unrelated).

We interpret the probability of the most likely sequence of an HMM for a sequence of observations as sequence similarity. Using this similarity, we classify the new documents, e.g., by taking the most probable HMM's document type or using a threshold on the similarity value.

Complementarity-based First, we use an rSCD matrix and classify truly unrelated documents and complementary documents into a single class *unrel*. Def. 2 detects complementary documents and thus allows to separate truly unrelated doc-

[b] `https://www.nltk.org/`
[c] `https://radimrehurek.com/gensim/`
[d] `https://numpy.org/`
[e] `https://www.scipy.org/`
[f] `https://pomegranate.readthedocs.org/`

uments from complementary documents. The implementation of Alg. 1 considers each pair of SCDs, i.e., SPO tuples, between $t_i^r \in g(d^r)$, $d^r \in \mathcal{D}_r$ and $t_j \in g(d')$. Due to the huge amount of pairs, we randomly sample 100 pairs from each set and consider each of their combinations. Then, we calculate all complementarity types $x \in \{\mathtt{s}, \mathtt{p}, \mathtt{o}, \mathtt{sp}, \mathtt{so}, \mathtt{op}, \mathtt{spo}\}$ for each pair t_i^r, t_j and return continuous values from the indicator function $\mathfrak{C}_x$. For each item in the SPO tuples of t_i^r, t_j, we use the tool Morphy to extract matching entities in WordNet. If there are multiple possible entities, we consider all possible entities and use the path similarity from WordNet to detect if the entities share the same concept or a common meaning. Entities with path similarities smaller than 0.1 are treated as different. Finally, we return the average or maximum across all path similarities as complementarity value $\mathfrak{C}_x$. For example, if the object of t_i is represented by the entities e_1, e_2 and the object of t_j by the entities e_1', e_2', the complementarity value max is given by $\max\{sim_{path}(e_1, e_1'), sim_{path}(e_1, e_2'), sim_{path}(e_2, e_1'), sim_{path}(e_2, e_2')\}$. We normalize the complementarity values after each sum of Definition 2 such that $\mathfrak{c}(d', d) \in [0, 1]$. Furthermore, we also calculate $\mathfrak{c}_x(d', d)$ only considering complementarity type x, i.e., $w_x = 1$ and $w_{x'} = 0 \forall x' \in \mathcal{X} \setminus x$.

Document Classification To classify documents as one of the five document types, we use Algs. 2 to 5. Using the corpus about car manufacturers, we train a cSCD and cSCD$^{\mathrm{f}}$ matrix on eight documents about car manufacturers and use eight cities as complementary documents during the training. As a baseline, we train an rSCD matrix on the same eight documents about car manufacturers. Additionally, we build corpora of eight documents for each of the document types, these five corpora are disjoint to the corpora used while training the matrices. We run four cycles of training HMMs and testing their classification performance. In each cycle the documents are randomly split into four documents for training and four documents for testing. We use the average across the results of the four cycles. As described for the complementarity-based approach in the previous paragraph, we use the path similarity of WordNet but return the average across all path similarities as the complementarity value $\mathfrak{C}_x$ in the set of relations $\mathcal{C}_x$.

6.3. *Results*

We present the result in three parts: First, we only compare the similarity and complementarity values of the differences document types. Second, we present the classification performance of documents using an rSCD, cSCD, and cSCD$^{\mathrm{f}}$ matrix. Finally, we illustrate which algorithms are needed by the techniques online and offline, including consequences on the runtime.

In Fig. 3, the similarity values and complementarity values are scaled to the interval $[0, 1]$. In the left plot, the sequence similarities gained from the similarity-based approach are shown for all five document types. The similarity value of *compl* and *unrel* documents is nearly equal. Thus, it is not possible to detect complemen-

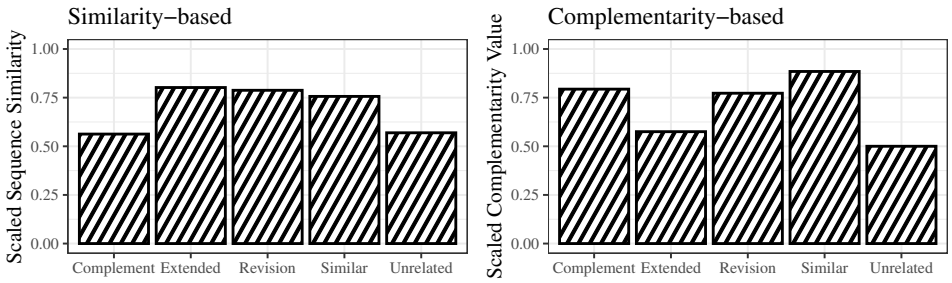


Fig. 3. Left: Average scaled similarity values per document type gained from the similarity-based approach using an rSCD matrix. Right: Average scaled complementarity values yielded by Def. 2.

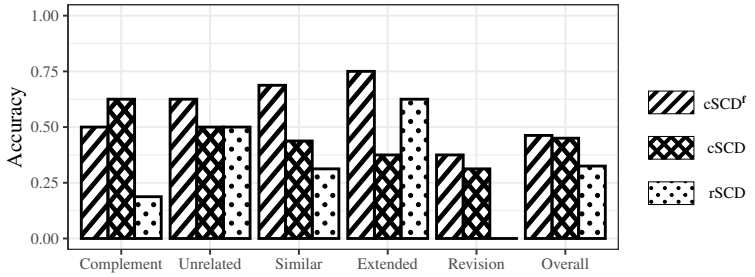


Fig. 4. Accuracy classifying each document type using a cSCD^{f}, cSCD or rSCD matrix.

tary documents using the similarity-based approach presented in [2]. In the right plot, the complementarity values max $\mathfrak{c}_{op}(d', d)$ are shown for all five classes. Complementary documents have a much higher value than unrelated documents, therefore it is possible to separate complementary documents from unrelated documents using a threshold $\theta_{\mathcal{D}}$ in Alg. 1. Interestingly, extended documents are nearly as complementary as unrelated documents and revisions are similar to complements using our definition of complementarity.

In Fig. 4, we compare the performance of an rSCD matrix with a cSCD and cSCD^{f} matrix. For all five document types and all three matrices the accuracy is shown. In all cases, the cSCD^{f} matrix results in the best values, except for complementary documents. Complementary documents are classified best by the cSCD matrix. Presumably, the cSCD matrix contains more relevant information about complementary documents than the cSCD^{f} matrix after the filtering. Overall, the cSCD^{f} matrix performs best, the rSCD matrix worst and the cSCD in between.

In general, the accuracy values in this five-classes setting are lower than the results gained by Kuhr et al. with four classes. Adding a fifth type of documents increases the difficulty of the classification problem. Randomly choosing one of five

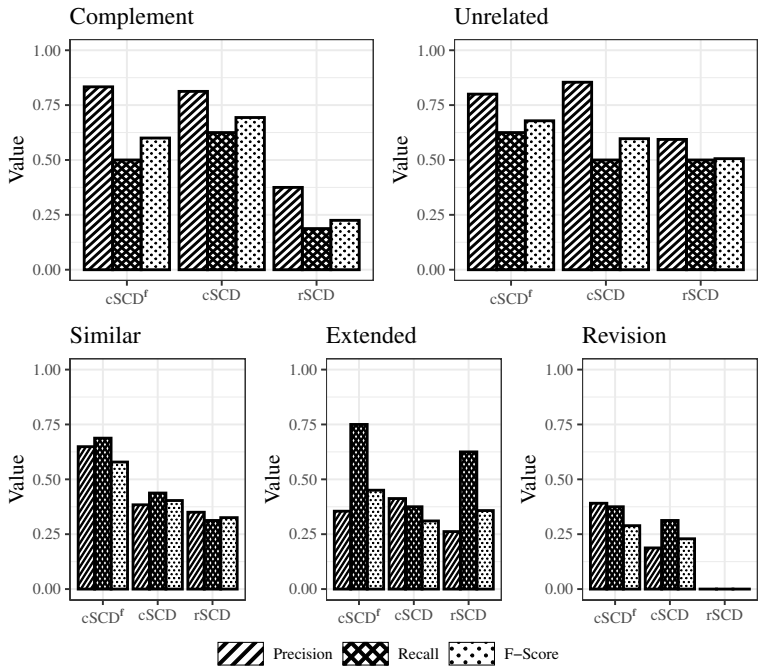


Fig. 5. Precision, recall and F1-Score classifying each document type using a $cSCD^f$, cSCD or rSCD matrix.

types would result in an accuracy of 0.2 while we reach accuracies between around 0.4 and 0.75. Compared to Kuhr et al., we use sentences, i.e., a form of a toppling window, instead of sliding windows over the text, which might also influence the accuracy as well. However, using the $cSCD^f$ (and cSCD) matrix improves the accuracy values significantly in our case study, especially compared to the rSCD matrix, which is the basis of the approach by Kuhr et al.

In Fig. 5, precision, recall and F1-Score are shown for each document type and matrix. Again, we notice that the cSCD matrix works best on the complement document type while the $cSCD^f$ matrix works best overall. Especially, similar, unrelated, and complementary documents are classified well. Classifying extended and revised documents seems to be more difficult because they consist of related, unrelated, and maybe even complementary sentences. For an HMM, it is difficulty to model the MPSCD similarities of a revised document, because the sequence may contain high positive, high negative, and even small values that do not follow any scheme. Working with sliding windows or larger amounts of documents would result in more data, which can be used to train the cSCD or $cSCD^f$ matrices and might lead to overall better results.

In Fig. 6, the algorithms needed by the techniques are shown. For each technique, the algorithms needed offline (training) and online (classifying a new document)

Technique	**Def. 2 only**		**rSCD**		**cSCD**		**cSCD^{f}**	
Algorithm	Offline	Online	Offl.	Onl.	Offl.	Onl.	Offl.	Onl.
Form rSCD matrix			✓					
Form cSCD matrix					✓		✓	
Filter matrix, cSCD^{f}							✓	
Estimate MPSCD			✓	✓	✓	✓	✓	✓
Train HMMs			✓		✓		✓	
Classify with HMMs				✓		✓		✓
WordNet Similarity		✓			✓		✓	

Fig. 6. The different algorithms used by the techniques marked if needed online or offline. Complement classification with Def. 2 only classifies *unrel* and *compl*; to classify between all five document types the algorithms of rSCD are also needed.

are listed. Regarding the runtime of the approaches, the calculation of the path similarities in WordNet are the most expensive part. Thus, the approaches using WordNet are much more expensive, especially if WordNet is used online during the classification. For example, calculating the values for the right plot of Fig. 3 has a total runtime of 2.25 hours and as we see by the check mark in the first column of Fig. 6, the calculations have to be done online. In contrast, forming the cSCD matrix used in the case study needs 41.1 hours, however, this is done only once and offline. Forming an rSCD matrix and training an ensemble of HMMs takes a couple of minutes. The classification used by rSCD, cSCD and cSCD^{f} matrices only quickly estimates MPSCDs and uses the pre-trained ensemble of HMMs.

In summary, using the definition of complementarity we are able to distinguish unrelated and complementary documents. However, the calculation of complementary SCDs is slow using WordNet. Thus, the cSCD and cSCD^{f} matrix combine corpora of complementary and related documents and allow a fast straightforward classification of all five document types.

7. Conclusion

If an agent is presented with a new document, it has to decide whether to extend its corpus with the new document or not depending on the document's type. The approach presented in this article enables the agent to classify a document into five types: similar, revised, extended, unrelated, and complementary. The approach operates on the SCDs of the new document and the agent's corpus during the classification. To this end, we first give a definition of complementary SCDs and define complementary documents based on their complementary SCDs. Second, we present the approach, which forms a combined SCD-word distribution matrix containing related and complementary SCDs for detecting complementary documents among the four other document types. In a case study, we demonstrate that the definition of complementarity allows an agent to separate unrelated and complementary documents. Additionally, we show that the performance using the combined SCD-word distribution matrix classifying documents of five types outperform previous

techniques not using combined SCD-word distribution matrices.

Our future work pursues the goal to enhance efficiency when checking for complementarity in a given taxonomy, which might be huge. Additionally, we are interested in associating more types of relations like complementarity between SCDs in SCD-word distribution matrices.

Acknowledgment

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

References

[1] F. Kuhr, T. Braun, M. Bender, and R. Möller, "To extend or not to extend? Context-specific corpus enrichment," in *AI 2019: Advances in Artificial Intelligence - 32nd Australasian Joint Conference, Adelaide, SA, Australia, December 2-5, 2019, Proceedings*, 2019, pp. 357–368. [Online]. Available: https://doi.org/10.1007/978-3-030-35288-2_29

[2] ——, "Augmenting and automating corpus enrichment," *Int. J. Semantic Comput.*, vol. 14, no. 2, pp. 173–197, 2020.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] T. Reuters, "OpenCalais," *Retrieved June*, vol. 16, 2008.

[9] 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.

[10] 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.

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

[12] 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.

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

[14] 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.

[15] Y. Zhang, "Prediction of financial time series with hidden Markov models," Ph.D. dissertation, Applied Sciences: School of Computing Science, 2004.
[16] N. Nguyen and D. Nguyen, "Hidden Markov model for stock selection," *Risks*, vol. 3, no. 4, pp. 455–473, 2015.
[17] 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.
[18] 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.
[19] 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.
[20] 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.
[21] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.
[22] G. Angeli, M. J. Johnson Premkumar, and C. D. Manning, "Leveraging linguistic structure for open domain information extraction," *Proceedings of the Association of Computational Linguistics (ACL)*, pp. 344–354, 2015. [Online]. Available: https://doi.org/10.3115/v1/P15-1034
[23] M. Bender, F. Kuhr, and T. Braun, "To Extend or not to Extend? Complementary Documents," in *16th IEEE International Conference on Semantic Computing, (ICSC 2022), Virtual, January 26-28*. IEEE, 2022, pp. 17–24.
[24] 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. University of California, Los Angeles: Academic Press, 1972, pp. 1–8.
[25] G. A. Miller, "WordNet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.