To Extend or not to Extend? Complementary Documents

Magnus Bender, Felix Kuhr
University of Lübeck
Institute of Information Systems
Ratzeburger Allee 160, 23562 Lübeck
{bender, kuhr}@ifis.uni-luebeck.de

Tanya Braun
University of Münster
Computer Science Department
Einsteinstr. 62, 48155 Münster
tanya.braun@uni-muenster.de

***Abstract*—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 focuses on complementarity, 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 referenced in the tuples, (ii) a problem definition and solution approach for classifying complementary documents, and (iii) a case study assessing classification performance for complementary documents.**

I. 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 on a corpus extension, the agent has to determine if a document is related to another document. 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 a distribution over a vocabulary, or SCD-word probability distributions, representing how often words appear around the locations of SCDs. Kuhr et al. [1], [2] have worked with three categories of documents based on similarity defined on SCD-word probability distributions: (i) quasi copies, a.k.a. similar documents, (ii) extensions, and (iii) revisions. All other documents are assumed unrelated. 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 *complementarity*, which unfortunately is hard to define given simply words or numbers in distributions or vector representations. Complementary documents 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.

To get a grip 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 entities 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 are able to formally define complementary documents and specify a corresponding document classification problem, thereby adding another document type – complement – to the set of document categories mentioned above. To solve the classification problem, we calculate a complementarity value between two documents based on their SCDs and how they interrelate given a taxonomy. Based on the complementarity value, an agent may decide to extend its corpus with a document or not. The agent may even use complementary SCDs to highlight points of interest, that is, areas where complementary SCDs occur frequently. Specifically, the contributions of this paper are:

(i) a definition of the document classification problem for complements and, for solving the problem, a definition of complementarity for SCDs in the form of relational tuples and a definition of complementarity for documents based on complementary SCD,
(ii) a solution approach to the problem, which can be used for corpus extension based on complementarity, and
(iii) a case study regarding the classification performance of our solution approach, compared against the corpus extension method of [1] using similarity.

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. Afterwards, we specify complementarity based on SCDs and present a solution approach to classifying complementary documents. Next, we present a case study and end with a conclusion.

II. 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] even automatically attach data from DBpedia [9] acting as external knowledge base (KB) to extractable named entity (NE) in the text. That is, the extractable NEs are matched to data in a KB to add additional data from the KB to the document. Adding additional 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 has 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].

III. Preliminaries

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

A. 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 $\mathcal{V}=(w_1,\ldots,w_V)$, $V \in \mathbb{N}$, and can be represented as a one-hot vector of length V having a value of 1 where $w = w_i$ and 0's otherwise.
- 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 $\mathcal{D}$ represents a set of $D \in \mathbb{N}$ documents $\{d_1,\ldots,d_D\}$ and $\mathcal{V}_\mathcal{D}$ returns the corpus-specific vocabulary containing all different words occurring in documents of corpus $\mathcal{D}$.
- 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 surrounding position ρ in d, i.e., $win_{d,\rho}=(w^d_{(\rho-i)},...,w^d_\rho,...,w^d_{(\rho+i)})$, $i \in \mathbb{N}$ and ρ marks the middle of the window.
- 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.

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

function BUILDMATRIX(Corpus $\mathcal{D}$)
  Input: $\mathcal{D}$
  Output: $\delta(\mathcal{D})$
  Initialize an $m \times V$ matrix $\delta(\mathcal{D})$ with zeros
  for each $d \in \mathcal{D}$ **do**
    for each $t \in g(d)$ **do**
      for ρ of t **do**
        for each $w \in win_{d,\rho}$ **do**
          $\delta(\mathcal{D})[t][w]$ += $I(w, win_{d,\rho})$
  Normalize $\delta(\mathcal{D})[t]$
  return $\delta(\mathcal{D})$

B. SCD-Word Probability Distributions

We define an additional representation for each SCD by taking a vector of length V, where $V = |\mathcal{V}_\mathcal{D}|$ s.t. each vector entry refers to a word in the vocabulary $\mathcal{V}$ of all documents in 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 in 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.

Algorithm 1 generates the SCD-word probability distribution for all m SCDs in the SCD set $g(\mathcal{D})$. Equation (1) represents the SCD-word probability distribution by an $m \times V$ matrix $\delta(\mathcal{D})$, with the SCD-word probability distribution vectors forming the rows of the matrix:

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

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. 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

Algorithm 2 Estimating MPSCDs

function ESTIMATEMPSCD(new document d', Number of SCDs M, matrix $\delta(\mathcal{D})$)
  Input: Document d', Number M, matrix $\delta(\mathcal{D})$
  Output: $\mathcal{W}$ containing MPSCDs t, MPSCD similarity values s, window details $win_{d',\rho}$
  $\sigma \leftarrow \frac{words(d')}{M}$, $\mathcal{W} \leftarrow \emptyset$
  for $\rho \leftarrow \frac{\sigma}{2}$; $\rho \leq words(d)$; $\rho = \rho + \sigma$ **do**
    $\delta(win_{d',\rho}) \leftarrow$ new zero-vector of length V
    for $w \in win_{d',\rho}$ **do**
      $\delta(win_{d',\rho})[w] += I(w, win_{d',\rho})$
    $t \leftarrow \arg\max_i \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d',\rho})|}$ in $win_{d',\rho}$
    $s \leftarrow \max_i \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d',\rho})|}$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(t, s, win_{d',\rho})\}$
  return $\mathcal{W}$

(MPSCD). Moving over all M windows, this procedure results in a sequence of MPSCDs, where each MPSCD is associated with a location in the new document. Algorithm 2 outlines the procedure, which not only returns the MPSCDs but also the cosine similarity values of each MPSCD and information about the windows.

C. Corpus Extension using Similarity

Using the sequence of 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.

IV. ESTIMATING COMPLEMENTARY DOCUMENTS

This section presents a new 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 document classification problem for complementary documents. Second, we provide a definition of complementary documents based on SCD similarity values to solve the problem. Third, we present an approach to classifying a new document d' as complementary

by analyzing the structure of SCDs associated with d' and those SCDs associated with documents in $\mathcal{D}$.

A. 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 to a binary classification problem s.t.

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

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 similarity values and defining a complement by using the notion of complementary SCDs. Based on these definitions, we specify a solution approach for corpus extension.

B. Complementary Documents

To classify a document as 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 presenting a definition of complementary SCDs and documents, let us consider an example for a new document d' containing complementary content to the content of documents in corpus $\mathcal{D}$. We will 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 in ξ or the predicates between the entities share a common meaning. Formally, the following seven types of complementarity between SCDs t_i and t_j exist:*

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

where, with $\uparrow$ referring to the concept in ξ that an entity belongs to, $o_i^\uparrow = o_j^\uparrow$, $s_i^\uparrow = s_j^\uparrow$, and p_i is a synonym that shares a common meaning with p_j. Let $\mathcal{X}$ refer to the set of the different complementarity types $\{\mathtt{s}, \mathtt{p}, \mathtt{o}, \mathtt{sp}, \mathtt{so}, \mathtt{op}, \mathtt{spo}\}$.

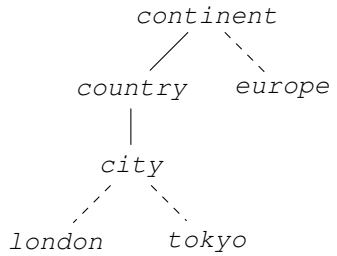
An indicator function $\mathfrak{C}_x(t_i, t_j)$, $x \in \mathcal{X}$, returns 1 if t_i and t_j fulfil the conditions mentioned above for x-complementarity and otherwise 0, including when t_i or t_j is not in SPO format.

Generally, it might be possible to adapt the return value of the indicator function $\mathfrak{C}_x(t_i, t_j)$ to include uncertainty. 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 to $g(d) = \{t_2, t_4\}$ 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 hierarchy, where solid lines represent the hierarchy between classes of a given taxonomy and dashed lines represent instances of classes from the taxonomy,



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 from `continent`, which is true for `europe`, too. Thus, we have a `o`-complementary for t_1 and t_3 and for t_2 and t_4.

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 fewer and fewer identical entities, with the top entry only requiring that the three positions are filled with different instances of the same concept, i.e., what falls

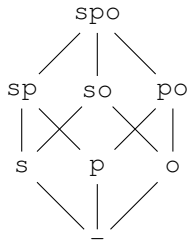


Fig. 1: The complementarity types of Def. 1 in a lattice.

under complementarity of lower levels also falls under complementarity of higher levels. Next, we define complementary documents based on Def. 1.

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

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

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

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

Given the complementarity lattice, it is reasonable to have non-zero weights only between complementarity types of the same level. E.g., given spo-complementarity, w_x should be set to zero for every other complementarity type, i.e., $\forall x \in \mathcal{X}, x \neq \texttt{spo}$, as these types x would be covered by spo. E.g., 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. 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 sense of complementarity, spo. Looking explicitly for complementary documents about a certain subject s, a high threshold and op-complementarity might be a fitting choice.

Given the decision criterion in Eq. (4), we solve the document classification problem of Eq. (2). Assuming that complements are considered a subset of unrelated documents, we could focus computing Eq. (4) for those documents that are otherwise classified as unrelated. How well this classification works, we showcase during the case study of Section V. Next, 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.

C. 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, which it may have previously classified as unrelated. To perform the task, the agent applies the definitions above for reaching a decision.

Algorithm 3 Corpus Extension with Complements

```
1: function EXTENDCOMPLEMENT(D, d', θ_D, {w_x}_{x∈X})
2:     if g(d') = ∅ then
3:         Add SCDs to d' using OpenIE
4:     c ← 0
5:     for each t_i ∈ g(d') do
6:         for each d ∈ D do
7:             for each t_j ∈ g(d) do
8:                 for each x ∈ X do
9:                     c ← c + w_x C_x(t_i, t_j)
10:    if c > θ_D then
11:        return true
12:    return false
```

Algorithm 3 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 considers 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.

D. Discussion

The following paragraphs discuss complements in the context of the document types mentioned above, as part of the task of document retrieval as well as for augmenting user output by returning positions of interest.

a) Complements as a Document Type: We can introduce a complement as another document type ($compl$) to the corpus extension setting in [1] and [2]. The classification problem there is defined given the sequence of MPSCD similarity values $\mathcal{W}$ computed by Alg. 2 for an unknown document d' and a corpus $\mathcal{D}$ as follows:

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

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 from [1] and [2] and then apply Alg. 3 to the unknown document if the previous classification returns $unrel$.

b) 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 complementary values. In contrast to Alg. 3, 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.

c) 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. 3 classifying a document as a complementary document. We denote those positions as *positions of interest*.

With this definition of complementarity in place and a solution approach specified with Alg. 3, we turn to evaluating the setting in a case study.

V. Case Study

In this section, we present a case study illustrating the potential of the definition of complementary documents. We demonstrate that document classification using the sequence of MPSCD similarity values as evidence in an HMM is not able to detect complementary documents. We show that an implementation of Alg. 3 performs well on the problem. Before we look at the results, we describe the corpus and workflow used in the case study.

A. 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[1]. 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 12 Wikipedia articles: *Apple Inc.*, *Apple*, *IPhone*, *Microsoft Windows*, *Google*, *Donald Trump*, *Atlantic Ocean*, *Angela Merkel*, *Baltic Sea*, *SpaceX*, *Lawyer*, and *Titanic*. 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.* 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.

B. Workflow and Implementation

Algorithm 2 in combination with the HMM-based classification and Alg. 3 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 [23] interface, provided by the Natural Language Toolkit[2], to detect if entities share the same concept or a common meaning in the sense of Def. 1.

Our implementation is optimized to run on multiple processor cores and uses the libraries Gensim[3], NumPy[4], SciPy[5] and Pomegranate[6]. 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-word matrix using Alg. 1, 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 Morphy use their own default preprocessing.

a) Corpus Extension using Similarity [2]: A new document may be classified belonging to one of the four classes sim, rev, ext or $unrel$. In this scenario, we assume truly unrelated documents and complementarity documents are both $unrel$ documents. The sequence of MPSCD similarity values is calculated for each new document and transformed into a sequence of observations. Then, the Viterbi algorithm [21] is used to calculate the probability of the most likely sequence

[1] https://w.wiki/4FUS

[2] https://www.nltk.org/

[3] https://radimrehurek.com/gensim/

[4] https://numpy.org/

[5] https://www.scipy.org/

[6] https://pomegranate.readthedocs.org/

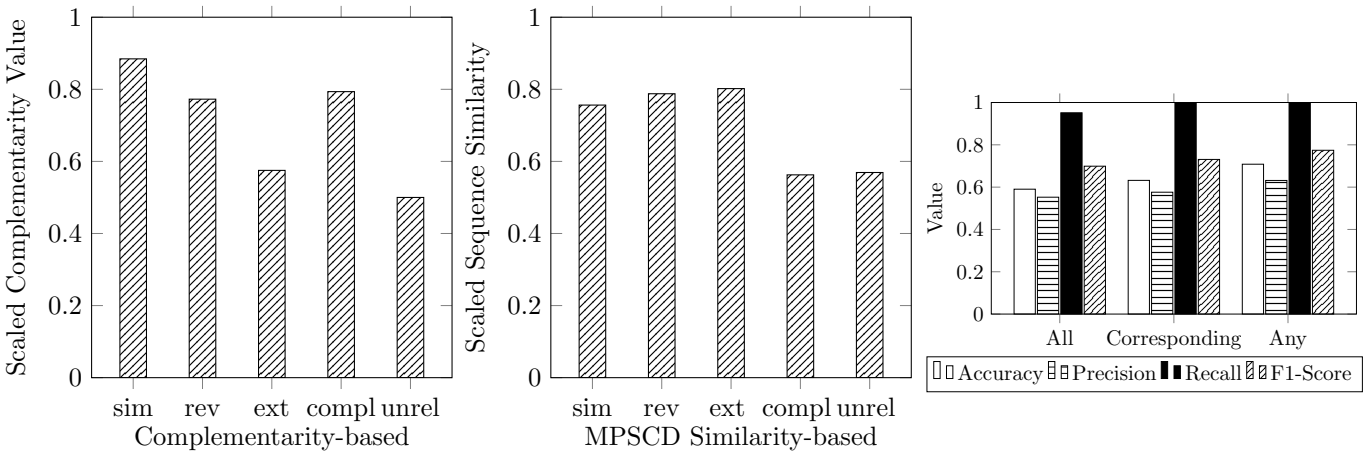


Fig. 2: Left: Average scaled similarity values per class gained from the MPSCD-based approach. Middle: Average scaled complementarity values per class. Right: Accuracy, precision, recall, and F1-score detecting complementary documents in the set of unrelated and complementary documents.

given the observations for each of the four HMMs, representing sim, rev, ext or $unrel$. We understand 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 class or using a threshold on the similarity value.

b) Corpus Extension using Complementarity: Using the similarity for corpus extension, we classify truly unrelated documents and complementary documents in a single class $unrel$. Def. 2 detects complementary documents and thus allows to separate truly unrelated documents from complementary documents. The implementation of Alg. 3 considers each pair of SCDs, i.e., SPO tuples, between $t_i \in g(d)$, $d \in \mathcal{D}$ 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, t_j and return continuous values from the indicator function $\mathfrak{C}_x$. For each item in the SPO tuples of t_i, t_j, we use the library 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.

C. Results

In Fig. 2, we present the results of the case study. The similarity values and complementarity values are scaled to the interval $[0, 1]$. In the middle plot, the sequence similarities gained from the MPSCD-based approach are shown for all five classes. The similarity value of $compl$ and $unrel$ documents is nearly equal. Thus, it is not possible to detect complementary documents using the MPSCD-based approach presented in [2]. In the left 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 $\theta_{\mathcal{D}}$. Interestingly, extended documents are nearly as complementary as unrelated documents and revisions are similar to complements using our definition of complementarity.

The MPSCD-based classification has a much smaller runtime than the complementarity-based classification approach. The following durations are measured excluding the runtime of OpenIE. Forming the SCD-word matrix and training the HMMs takes on the corpus used in the case study 2.5 minutes and the classification of all documents takes 1.7 minutes. Together, 4.2 minutes are needed by MPSCD-based classification of which 2.5 minutes may be done offline. The complementarity-based classification approach does not need any training and reaches a total runtime of 2.25 hours on the corpus used in the case study. Thereof, the calculation of the path similarities in WordNet are the most expensive part. To reduce the runtime it would possible to use a smaller sample size than 100, since the sample size influences the runtime quadratically. Small sample sizes do not seem to significantly worsen the results.

In the right plot of Fig. 2, the classification performance of Alg. 3 is shown. In terms of an agent's workflow, we assume that first the document type detection algorithm from [1] and

[2] is applied to a set of unknown documents and afterwards, Alg. 3 is applied to those documents where the previous classification returns $unrel$. The idea is that an agent does not need to perform complement detection on all documents that have to be classified, but only on those documents that are classified as unrelated.

We use three different scenarios to evaluate the classification performance for complementary documents. To be classified as complementary document, the document is detected as complementary document to:

(i) all documents in the corpus (*all*),
(ii) exactly its corresponding document in the corpus, i.e., the city of the car manufacturer's headquarter to the car manufacturer (*corresponding*), and
(iii) any document in the corpus (*any*).

In general, it depends on the agent which of the three scenarios promises the most added value for its task. The corpus-specific thresholds used in the right plot of Fig. 2 are $\theta_{\mathcal{D},all} = 0.675$, $\theta_{\mathcal{D},corresponding} = 0.65$ and $\theta_{\mathcal{D},any} = 0.4$.

In all three scenarios the recall is very high, such most complementary documents are detected. The scenario *any* reaches the best scores as it is more robust against document outliers in the corpus, e.g., the unrelated document *Apple Inc.* containing a paragraph about an autonomous car.

In summary, the case study shows that the previous HMM-based classification method has its limits in terms of supporting an agent in choosing documents for corpus extensions that provide complementary information. Algorithm 3 provides a good foundation to find complementary documents for an agent's corpus.

VI. Conclusion

If an agent is presented with a new document, it may decide whether to extend its corpus with the new document or not by classifying the document in a context-specific way. This paper enables the agent to detect complementary documents to avoid being stuck a bubble of similarity. We argue, complementary documents present the agent's context from a different point of view or provide further information. The approach operates on the SCDs of the new document and the corpus. This paper gives a definition of complementary SCDs and defines complementary documents based on their complementary SCDs. To extend a corpus based on complementary documents, a new document is checked for complementarity, possibly after is has first been classified as unrelated based on its similarity to the corpus. In a case study, we demonstrate that our definition of complementarity allows an agent to separate unrelated and complementary documents, while an MPSCD similarity-based approach can not distinguish complementary and unrelated documents.

Our future work pursues the goal to associate a document with complementary SCDs. Then, it would be possible to form a complementary SCD-word distribution matrix for a corpus and detect complementary documents without using WordNet. Additionally, omitting WordNet would circumvent the time-consuming path similarity calculations, making it also an interesting point for further research.

Acknowledgment

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

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] G. A. Miller, "WordNet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.