Augmenting and Automating Corpus Enrichment

Felix Kuhr
University of Lübeck
Institute of Information Systems
Ratzeburgerallee 160, 23562 Lübeck
kuhr@ifis.uni-luebeck.de

Tanya Braun
University of Lübeck
Institute of Information Systems
Ratzeburgerallee 160, 23562 Luebeck
braun@ifis.uni-luebeck.de

Ralf Möller
University of Lübeck
Institute of Information Systems
Ratzeburgerallee 160, 23562 Luebeck
moeller@ifis.uni-luebeck.de

***Abstract*—An agent in pursuit of a task may work with a reference library containing documents with linked subjective content descriptions. Faced with a new document, an agent has to decide whether to include the new document in its reference library. Basing the decision on only words, topics, or entities has shown not to lead to a balanced performance for varying documents. Even a combination of words and descriptions does not lead to a single indicator, requiring manual post-processing. Therefore, in this paper, we build a single indicator by detecting the type of a new document using sequential information about descriptions, thus automating the decision. Specifically, an ensemble of hidden Markov models for the document types detects the type of a document. The agent then bases its decision on the detected type. Using hidden Markov models also allows for identifying positions of interest within a new document. A case study shows the effectiveness of our approach.**

I. Introduction

An agent in pursuit of a task, explicitly or implicitly defined, may work with an individual set of documents (corpus) as a reference library. A person assembling a set of scientific articles as related work describes such a setting, with the person as the agent, the compiling of articles as the task, and the articles as the corpus. From an agent-theoretic perspective, an agent is a rational, autonomous unit acting in a world, perceived through sensors, fulfilling a task, e.g., an agent providing document retrieval services given requests from users. The documents in the corpus may be annotated with subjective content descriptions (SCDs) to improve the agent's performance, i.e., SCDs are relevant to an agent's task. SCDs can be notes, automatically or manually appended to specific sentences of an article, that may provide explanations, references, or additional data. Thus, SCDs add data relevant for the agent's task, and that data has a connection to specific locations in the document. But, what should an agent do if presented with a new document, which typically has no SCDs? Without having thoroughly processed the new document, the question for the agent is: Does that document have anything of value to add in the context of the given corpus? We call extending a corpus with a document that adds value corpus enrichment. Kuhr et al. [1] have introduced an approach for corpus enrichment by estimating most probable SCDs (MPSCDs) for new documents considering the composition of documents in a corpus and the document-specific set of SCDs. The investigated features in [1] do not provide a single indicator for the decision and manual post-processing is necessary.

Therefore, we turn to fully automating corpus enrichment. To this end, we consider document types and decide corpus enrichment based on the document type detected. We extend the approach in [1] by considering the sequence in which SCDs appear in a document. For each document type, we learn a hidden Markov model (HMM) where each slice concerns a window in a document and the hidden state concerns whether the SCD in the window is related to the task. Together, the HMMs for all types form an ensemble. Given a new document, we use the Viterbi algorithm [2] on each HMM to find the most likely sequence of (un)relatedness and its probability. The document type is then given by the type of the HMM that has the sequence with the highest probability. Based on this sequence, an agent can identify positions of interests (PoIs). Identifying PoIs augments corpus enrichment with providing location-specific information about new data. If the agent decides to extend its library with the new document based on the document type, it can even choose to retain initial SCDs, possibly adapting them, and then use them as a basis for enriching SCDs within the corpus, in an automatic [3], [4] or manual way.

Specifically, the contributions of this paper are: (i) solving the problem of identifying the type of a new document by comparing the probabilities of the most likely sequences of MPSCD similarity values generated from the different HMMs learned for each type, (ii) providing a decision making procedure for corpus enrichment based on the document type, and (iii) a case study regarding document type detection learning for three different corpora the respective four document type specific HMMs. In this paper, we focus on the following four types of documents: (i) similar documents, (ii) extensions of existing documents, (iii) revisions of existing documents, and (iv) unrelated documents. Using an ensemble allows for adding further HMMs to the ensemble if other document types become relevant. The ensemble also enables an agent to weight individual HMMs if certain document types are more interesting compared to others.

The remainder of this paper is structured as follows: We start with a look at related work. Then, we specify notations and recap estimating MPSCDs. Next, we present HMM-based document type detection and the decision making procedure for corpus enrichment. We also discuss finding PoIs in a most-likely sequence, followed by a case study. The paper ends with a conclusion and future work.

II. Related Work

Over the past 20 years, a considerable number of automatic (semantic) annotation systems have been developed. The systems extract named entities from text of documents and add so-called *semantic annotations* from externally available common-sense knowledge bases, enriching the documents with machine-processable data. Some well-known automatic annotation systems are YEDDA [5], MINTE [6], OpenCalais [7], YAGO[8], KDTA [9], or GATE [10]. However, the output of available automatic annotation systems have only very little agreement [11]. Some well-known sources of common-sense knowledge are DBpedia [12], NELL [13], and KnowledgeVault [14]. Named entities represent the link between documents and common-sense knowledge and link prediction is used to identify semantic annotations for an entity. Generally, link prediction describes the task of estimating the likelihood of a link (relation) existing between nodes (entities), given the links, and attributes of nodes within a graph [15]. Annotation systems aim at developing a knowledge graph augmented with data from external sources. The systems efficiently solve their underlying problem. However, we investigate a different problem, deciding if a new document provides value to an agent by working with SCD. We consider the context of documents and their content instead of focussing on external data from common-sense knowledge bases.

Surveying methods of text mining, one can base a decision on different aspects, e.g., (i) similarity of text in the spirit of tf.idf [16], 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) [17], comparing an estimated topic distribution of a new document with topic distributions of the documents in the corpus, or (iii) entity matching [18] using named-entity recognition (NER), comparing entities (and relations) retrieved from the new document with entities (and relations) of the SCDs in the corpus. All three approaches carry drawbacks: The first two, based on bag-of-words, ignore SCDs and the order of words. Additionally, they make it difficult to model that a document has to add value, i.e., not be a rephrased copy of an existing document or contain only unrelated data. The last approach has the problem that NER tools might not output annotations in the context of the task, which may lead to very few matches with SCDs of the corpus. Additionally, the decision in each case is a one-dimensional decision, based on one feature of the documents. Therefore, we aim at providing an approach to automatically make a multi-dimensional decision that considers the context of the task.

Another class of related work deals with HMM-based classification. Classification and statistical learning by HMMs has achieved remarkable progress in the past decade. Using a HMM is a well-researched stochastic approach for modeling sequential data, and it has been successfully applied in a variety of fields, such as speech recognition [19], character recognition [20], finance data prediction [21], [22], credit card fraud detection [23], and workflow mining [24]–[26].

III. Preliminaries

This section specifies notations and gives a brief overview of MPSCDs we use as observations in the HMM.

A. Notation

We define the following terms to formalize the setting of a corpus containing documents, each document associated with a type label and a repository of additional data, i.e., SCDs.

- A word w is a basic unit of discrete data from a vocabulary $\mathcal{V}=(w_1,\ldots,w_V)$. Each w is represented as a unit-basis vector of length V that has a value of 1 where $w=w_v$ and 0's otherwise.
- A document d is a sequence of words $(w_1,\ldots,w_D)$ where each w_d is from $\mathcal{V}$. The expression $words(d)$ refers to the number of words in d.
- A corpus $\mathcal{D}$ is a set of N documents $\{d_1,\ldots,d_N\}$.
- For each document $d\in\mathcal{D}$ exists a document-specific repository g containing a set of SCDs $\{(t_j,\{\rho_i\}_{i=1}^{l})\}_{j=1}^{s}$. SCDs can take any form. As such, their formats may be highly diverse. A standardized format would be the Resource Description Framework (RDF) but, for our main contributions, the specific format is irrelevant. Each SCD t is associated with a set of positions $\{\rho_i\}_{i=1}^{l}$ in document d where ρ_i refers to the ρ_i'th word in d. Given a document d or repository g, the terms $g(d)$ and $d(g)$ refer to the linked repository and document, respectively. The set of all SCDs of documents in $\mathcal{D}$ is given by $g(\mathcal{D})=\bigcup_{d\in\mathcal{D}} g(d)$.
- A new document d' has a specific type $type(d')\in\mathcal{T}$, $\mathcal{T}=\{d'_{sim},d'_{ext},d'_{rev},d'_{unrel}\}$, representing the following four types: 1) similar document, 2) document extension, 3) revision of a document, and 4) and unrelated document, respectively. The type of d' is determined with respect to the documents in $\mathcal{D}$.
- An SCD window $win_{d,t,\rho}$ refers to the words in d that surround the position ρ of $t\in g(d)$, i.e., $win_{d,t,\rho}=(w_{(\rho-i)},...,w_{\rho},...,w_{(\rho+i)})$, $i\in\mathbb{N}$ if ρ marks the middle of the window. The position of a word $w\in win_{d,t,\rho}$ is given by $pos(w,win_{d,t,\rho})$ (0-based numbering). The size of $win_{d,t,\rho}$ is given by $s(win_{d,t,\rho})$, i.e., $s(win_{d,t,\rho})=2i+1$ if ρ marks the middle of the window.
- Each word $w\in win_{d,t,\rho}$ is linked to an influence value $I(w,win_{d,t,\rho})$. The closer a word w is positioned to the position ρ of t, the higher is the influence value $I(w,win_{d,t,\rho})$. The influence value $I(w,win_{d,t,\rho})$ of w is given by the probability of the Binomial distribution at position $pos(w,win_{d',t,\rho})$, i.e.,

$$I(w,win_{d,t,\rho})=\binom{n}{k}\cdot\pi^{k}\cdot(1-\pi)^{n-k}, \qquad (1)$$

  where $n=s(win_{d',t,\rho})-1$, $k=pos(w,win_{d',t,\rho})$, and $\pi=\frac{\rho}{n}$, i.e., $\pi=0.5$ if t is at the center of $win_{d,t,\rho}$ and influence values to the left and right of ρ should be symmetric. The binomial distribution yields a probability for each word $w\in win_{d,t,\rho}$ that is higher the closer w is to the position of t.

B. Subjective Content Descriptions

SCDs are assumed to generate the words in the documents [1]. An SCD can be represented with a vector of length n, where $n = |\mathcal{V}(\mathcal{D})|$ and each vector entry refers to a word in $\mathcal{V}(\mathcal{D})$. The entry itself is a probability that describes how likely it is that the corresponding SCD generates the word, yielding an SCD-word distribution for each SCD. The SCD-word distribution for all m SCDs in $g(\mathcal{D})$ are represented by an $m \times n$ matrix $\delta(\mathcal{D})$, with the SCD-word distribution vectors forming the rows of the matrix:

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

To fill $\delta(\mathcal{D})$ given $\mathcal{D}$ using a maximum-likelihood strategy, one counts for each SCD t the number of occurrences of each word w in the windows $win_{d,t,\rho}$ of t over all documents and all positions. The occurrences are weighted by their influence value $I(w, win_{d,t,\rho})$. At the end, the rows are normalized to yield a distribution again. See [1] for details.

For a new document d', one can use $\delta(\mathcal{D})$ to estimate how well the existing SCDs may generate the words in d'. To do so, one computes $M \ll m$ MPSCDs for d' by building a vector representation $\delta(win_{d',t,\rho})$ of the words in the M windows in d' and finding the SCD that has a vector representation most similar to $\delta(win_{d',t,\rho})$. The SCD t most similar to $\delta(win_{d',t,\rho})$ (w.r.t. the cosine similarity) is given by:

$$\arg\max_{t} \frac{\delta(\mathcal{D})[t] \cdot \delta(win_{d',t,\rho})}{|\delta(\mathcal{D})[t]| \cdot |\delta(win_{d',t,\rho})|}. \quad (3)$$

Algorithm 1 describes estimating MPSCDs for new document d' using $\delta(\mathcal{D})$ given M. The output of Alg. 1 includes the set of MPSCDs $g(d')$, one for each window, as well as a set of tuples of similarity value and window for each MPSCD. Again, see [1] for details.

Algorithm 1 Estimating MPSCDs

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

IV. Automatic Corpus Enrichment

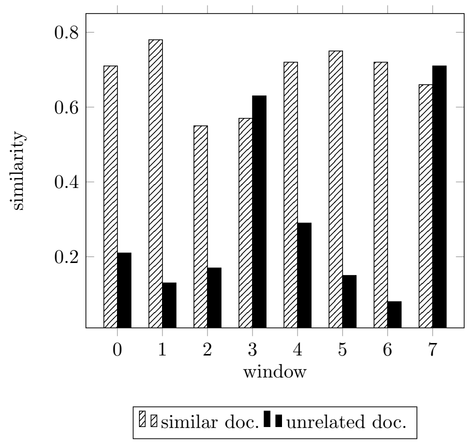
This section presents an approach to automatic corpus enrichment given a new document considering the documents in a corpus. We introduce the document type detection problem for a new document d' given an individual composition of documents, and present an HMM-based solution to estimate the probability of the most likely sequence of hidden states from the observable MPSCD similarity values in d'. We then provide a decision making procedure and describe how to find PoIs in d', augmenting corpus enrichment. Before going into details about document type detection, we take a look at the output of Alg. 1, specifically at the similarity values, to show how document type detection using the sequence of similarity values actually has a chance at success.

A. Document Types and their Similarity Values

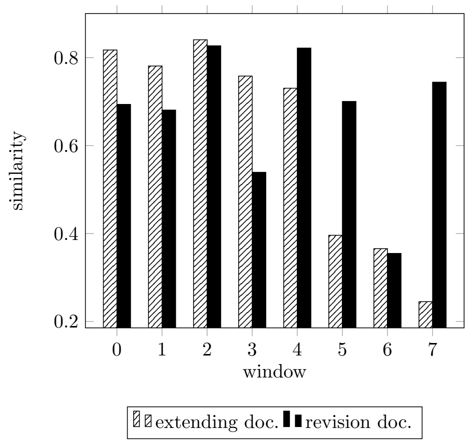
We assume that a document is labelled with a document type and we follow the idea of [1] that the type of a document depends on the MPSCD similarity values.

In Fig. 1, we present similarity values for documents of the four different document types, $d'_{sim}, d'_{ext}, d'_{rev}$, and d'_{unrel}. The values on the x-axis represent the SCD window number in a document, and the values on the y-axis represent the MPSCD similarity values. Generally, the similarity values change slightly between neighbouring windows for both similar and unrelated documents. But, the MPSCD values of similar documents are considerably higher compared to unrelated documents (Fig. 1a). Some MPSCD similarity values of unrelated documents can be higher than MPSCD similarity values of similar documents, e.g., in the third window and seventh window. Extending documents have high similarity values in the beginning of the document followed by windows with smaller similarity values. Revision documents have a significant change in the similarity value of some neighbouring windows (Fig. 1b), namely, those windows containing new and unrelated content. Based on these observations, we can describe the four types we consider in this paper as follows:

- Similar documents (d_{sim}): Content of a new document is similar to the content of a subset of documents in $\mathcal{D}$, i.e., new document tells about the same event, same persons, or topics. The values in the MPSCD similarity sequence are mostly in the category y_h or y_m.
- Extensions (d_{ext}): The content of a new document is mostly similar to the content of documents in $\mathcal{D}$ but contains additional content, semantically unavailable in any other document of corpus $\mathcal{D}$, i.e., the new document is an extended version of another document in corpus $\mathcal{D}$. Thus, the similarity values of the first $t-1$ observable MPSCD values are mostly in category y_h and y_m, because the content is similar. At some point, the similarity values reduce and are mapped mostly to y_l.
- Revisions (d_{rev}): The new document represents a revision of another document in $\mathcal{D}$. Thus, the content of the new document contains modified content generated by appending, replacing, or removing content. The values in



(a) Similar and unrelated documents.



(b) Revisions and extensions of documents.

Fig. 1: Representation of MPSCD similarity values of four documents of different document types.

the MPSCD similarity sequence are mostly in category the y_m and y_h, but also few of them are in the category y_l, namely, those values representing the modified content.

- Unrelated documents (d_{unrel}): The content of a new document is unrelated to the content of all documents in $\mathcal{D}$. The values in the MPSCD similarity sequence are in y_l or y_m but some similarity values can be in y_h.

Next, we define the document type detection problem and present an approach to solve it using an ensemble of HMMs.

B. Document Type Detection Problem

After estimating the MPSCDs of a new document d' using Alg. 1, we have a sequence of similarity values over the windows in d'. Generally, the document type detection problem asks for the most probable type of a new document d' given this observable sequence of similarity values:

$$\arg\max_{t\in\mathcal{Y}} P(t|\mathcal{W}), \tag{4}$$

As described in the previous section, the idea is that if the contents in the window are (un)related to the agent's task, then (low) high similarity values occur, which may vary over the course of a document, with known and unknown parts mixing. The consequence of this consideration is that we have a sequence of hidden states, encoding the relatedness of the document's content, and a sequence of observations as similarity values. Together, we can model this setup as an HMM. Therefore, we solve the document type detection problem by learning HMMs, one for each document type, determining the most likely sequence of hidden states for d', and then returning the type associated with the HMM with the highest probability of its sequence.

C. An Ensemble of HMMs

To detect document types, we form an ensemble of HMMs, one HMM for each document type, which we define next.

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

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

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

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

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

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

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

For a set of document types $\mathcal{T}$, an ensemble of HMMs $\mathcal{H}$ contains an HMM for each document type $type \in \mathcal{T}$. All $\lambda \in \mathcal{H}$ have the same Ω and Δ but different A and B.

For a new corpus, A and B are unknown and have to be learned. One famous technique for learning both matrices of an HMM is the Baum-Welch algorithm [27], which is a special case of the well-known EM algorithm [28]. Using a set of documents of a specific type, one can calculate for a hold-out set MPSCDs and their similarity values and train the HMM on this information using the Baum-Welch algorithm.

The discrete observation alphabet Δ requires discretizing similarity values. As a function $f : [0, 1] \mapsto \Delta$, it maps a

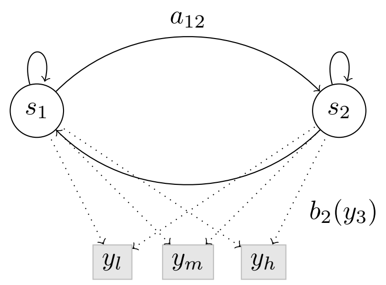


Fig. 2: Hidden Markov model containing two hidden states $\{s_1, s_2\}$ emitting three observation symbols $\{y_l, y_m, y_h\}$.

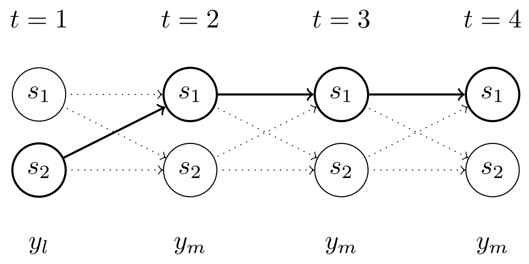


Fig. 3: Trellis of $O = (y_l, y_m, y_m, y_m)$, leading to the following most likely sequence of hidden states: (s_2, s_1, s_1, s_1).

similarity value x to one of the m symbols in Δ:

$$f(x) = \begin{cases} y_1 & 0 \leq x < th_1 \\ y_2 & th_1 \leq y < th_2 \\ \vdots & \\ y_m & th_{m-1} \leq y < 1 \end{cases} \tag{5}$$

Algorithm 2 shows pseudocode, mapping the similarity values in $\mathcal{W}$ for a new document d' to the observation symbols in alphabet Δ based on function f, returning a sequence of observable symbols O. Generally, the discretization and its thresholds depend on the task of an agent and can be adapted to each problem individually. We follow the idea of Kuhr et al. [1], using $m = 3$, and select $th_1 = 0.3$ and $th_2 = 0.7$ such that $f(x)$ maps MPSCD values to y_l, y_m, and y_h, referring to low, medium, and high values. Figure 2 contains a graphical representation of a hidden Markov model $\lambda = (a_{ij}, b_j, \pi)$ containing observation symbols $\{y_l, y_m, y_h\} \in \Delta$, and the two hidden states $\{s_1, s_2\} \in \Omega$.

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

D. Detecting the Type of a New Document

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

Algorithm 2 MPSCD Similarity Discretization

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

Algorithm 3 Document Type Detection

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

The task of determining which sequence of variables is the underlying source of some sequence of observations is called a *decoding task*. We can calculate for each model the most likely sequence of hidden states using the Viterbi algorithm [2]. The Viterbi algorithm makes use of the dynamic programming trellis for computing the most likely hidden state sequence S for an observation sequence O. Before presenting a complete specification of the document type detection approach, let us consider an example of a most likely sequence given a set of MPSCD similarity values.

Example 1. *Assume that we have calculated a set of MPSCDs for a new document d' by using Alg. 1 and the corresponding MPSCD similarity values are as follows:*

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

Using Alg. 2 leads to the following observation sequence:

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

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

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

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

Algorithm 3 describes how to estimate the most probable type of a document using an ensemble of HMMs. The input parameters are given by the MPSCD values in $\mathcal{W}$ and the ensemble $\mathcal{H}$. In line 2 and 3, Alg. 3 initializes a temporary variable p and the output tuple s. In line 4, Alg. 3 calls Alg. 2 to set the observation sequence O to the sequence of the similarity values in $\mathcal{W}$ discretised based on function f as in Eq. (5). Afterwards, Alg. 3 iterates over the different HMMs in $\mathcal{H}$. In each iteration, the algorithm calculates a most probable sequence S given O using the Viterbi algorithm (line 6). In line 8 to 10, Alg. 3 tests if the probability $prob(S)$ of the current most likely sequence S is higher than the previously seen highest probability. If the probability is higher, Alg. 3 saves the current most likely sequence S and the HMM λ in s. After iterating over each HMM in $\mathcal{H}$, Alg. 3 outputs the most likely sequence that has exhibited the highest probability among all most likely sequences as well as the corresponding HMM. This output we then use as a basis for the decision regarding corpus enrichment. Before moving on to decision making, let us consider an example for how document types and sequences may interact.

Example 2. *Assume that we have an unrelated document with similarity values that have led to the following observation sequence as well as the most likely sequence as an output of Alg. 3, meaning the HMM learned on unrelated documents has produced the most likely sequence with the highest probability:*

$$O_{unrel} = (y_l, y_m, y_l, y_l)$$
$$S_{unrel} = (s_2, s_2, s_2, s_2)$$

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

$$O_{ext} = (y_h, y_m, y_l, y_l)$$
$$S_{ext} = (s_1, s_1, s_2, s_2)$$

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

E. Automatic Decision Making: Corpus Enrichment

In this paper, we reduce corpus enrichment to making a decision based on the estimated type of a new document d'. Given a corpus $\mathcal{D}$ with SCDs $g(\mathcal{D})$, the agent has to perform the following steps *offline*:

(i) Build an SCD-word matrix $\delta(\mathcal{D})$ based on $g(\mathcal{D})$.
(ii) Train an ensemble of HMMs $\mathcal{H}$ using $\mathcal{D}$ and $\delta(\mathcal{D})$.
(iii) Specify function f for discretizing similarity values.
(iv) Specify which document types are relevant for the task and the current condition of corpus $\mathcal{D}$.

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

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

(i) Use Alg. 1 to estimate MPSCDs for d'.
(ii) Use Alg. 3 to estimate the document type $type$ of d'.
(iii) Based on $type$ and its own specification, add/forgo d'.

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

F. Augmenting Enrichment: Positions of Interest

Detecting the type of a new document can help an agent making a decision on extending a corpus with a new document or not. However, the agent might be interested in more details and not only in the type of a document. Let us assume that the agent is interested in extending a given corpus $\mathcal{D}$ with new documents that are classified as unrelated documents but share at least one section of related content with one other document. The output of document type detection (Alg. 3) includes the most likely sequence of hidden states, representing for each SCD window if the content is related or unrelated to the content of documents in the corpus. We can use the sequence of hidden states to get information about the content similarity in a SCD window instead of only having information about the similarity values of the MPSCD relating to a specific window. If a new document d' is unrelated to other documents in the given corpus $\mathcal{D}$ but parts of the document contain related content the agent might be interested in the parts containing related content to identify if document d' might contain new content relating to an already-known topic. We define these parts as PoIs in a document.

Identifying PoIs for d' requires document type detection using Alg. 3. Depending on the document type, the agent might be interested in different positions within the new document. Thus, we define the following positions of interest.

- For documents of type d_{sim}, the sections containing unrelated content are PoIs.
- Documents of type d_{ext} contain related content in the beginning of the document, which changes to unrelated content when the document extension starts. The position of the change is the PoI in d_{ext}.
- For document of type d_{rev}, the PoI is given by those positions containing modified (unrelated) content.
- For documents of type d_{unrel}, the sections containing related content are PoIs.

Detecting positions of interest in sequences relates to pattern recognition [29]. Next, we present a case study.

TABLE I: Document type detection performance considering four document types and three corpora.

	city corpus			president corpus			university corpus		
Document Type	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
d_{sim}	0.72	0.65	0.68	0.77	0.71	0.74	0.70	0.71	0.70
d_{unrel}	1.00	1.00	1.00	1.00	0.96	0.98	1.00	1.00	1.00
d_{ext}	0.93	0.86	0.89	0.91	0.84	0.87	0.86	0.92	0.89
d_{rev}	0.70	0.41	0.52	0.72	0.58	0.64	0.68	0.51	0.58

V. Case Study

After introducing the document type detection approach, we present a case study illustrating the potential of document type detection using the sequence of observable MPSCD similarity values as evidence in HMMs. We use the following three corpora within the case study, each containing articles from the free Wikipedia encyclopedia:

(i) Cities in Europe: documents about the largest cities in Europe (https://bit.ly/2kOvmwD).
(ii) US presidents: documents about presidents of the U.S. between 1789 and 2017 (https://bit.ly/2Z1v1G9), and
(iii) Universities: documents about the state and territorial universities in the U.S. (https://bit.ly/2mOzXAW).

A. Preprocessing

We work with a Java-based implementation of Alg. 1 and use the results as input parameters of our Python-based implementation of Alg. 2 and Alg. 3. Initially, we download all necessary articles from Wikipedia for the three corpora and preprocess the articles by: (i) lowercasing all characters, (ii) stemming the words, (iii) tokenizing the result, and (iv) eliminating tokens from a stop-word list of 337 words.

For each corpus and each type of document, we then generate a training set and test set in in the following way:

- Similar Documents (d_{sim}): We choose 90% of the corresponding Wikipedia articles as training data, i.e., 90% of the articles represent documents in the corpus. The remaining 10% of articles from the same corpus are used as a held-out set acting as the test data.
- Extending documents (d_{ext}): For each corpus, we choose the training set by selecting the corresponding Wikipedia articles and extend them with text from articles of the other two corpora. We proceed the same way to generate documents for the test set.
- Revision documents (d_{rev}): We use the *view history* function in Wikipedia. A current article represents a revision of an older version of the same article. Current articles contain modified sentences and additional content. We proceed the same way for generating the test set.
- Unrelated documents (d_{unrel}): We choose the training set by randomly selecting 90% of the Wikipedia articles from one corpus and randomly selecting articles from the two other corpora to fill the test set.

We have chosen a ratio of 90:10 between the size of the training set and the test set because of the size of documents in each corpus. However, we have performed additional evaluations adjusting the ratios up to 75:25, leading to similar results.

B. Document Type Detection

For each type of document, we take its training set and learn the transition and emission function of the HMM using the Baum-Welch algorithm in each corpus. This process results in four HMMs differing from each other in both the transition probabilities and emission probabilities. Each HMM represents exactly the configuration for one type of document. Now, we use Alg. 3 to perform the document type detection of documents in the test set. We use precision and recall to evaluate the performance of Alg. 3 instead of accuracy, since the application of accuracy is only meaningful for symmetric datasets, where false negatives and false positives counts are close, and false negatives and false positives have similar costs.

True positives (tp) refer to the number of documents whose document types have been correctly estimated, false positives (fp) to the number of documents whose document types have been falsely estimated, and false negative (fn) to the number of documents classified with a type of a document that was not found. Based on these values, we can evaluate the performance of Alg. 3 using precision and recall:

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

C. Results

We conduct experiments to test the performance of the document type detection approach based on the earlier described technique to generate corpora and the evaluation approach. Table I represents the performance of Alg. 3 using both, precision and recall for all three corpora. Generally, document type detection performs best on unrelated and extending documents. The performance on similar documents and revisions is worse than on unrelated and extending documents. For all three corpora, it is not easy to distinguish similar documents from the revision of a document. The transition probability and emission probability are similar for both types of documents in the corresponding HMMs. We have never classified revisions and similar documents as unrelated documents or document extensions, respectively, but sometimes we have classified documents of type d_{sim} as documents of type d_{rev}, and vice versa. We assume an agent interested in similar documents might accept document revisions because they are similar to the documents in a corpus. If an agent is only interested in similar documents or revisions, it must analyse documents of both types in detail, e.g., by using positions of interest. Generally, there exists no trivial approach in finding a solution to distinguish both types, because documents from both types behave similar, which leads to similar probabilities in their underlying HMMs. Kuhr et al. [1] have had the same problem in distinguishing a new document from type d_{sim} and d_{rev}. Their introduced indicators were the same for similar and revised documents in the president corpus, and only one of the five indicators is different for both types in the city corpus.

VI. Conclusion

If an agent is presented with a new, unknown document, this paper enables it to decide whether to extend its reference library with the new document or not by estimating its document type in a context-specific way. Specifically, we present how to model a window sequence with hidden Markov models to find a most probable sequence of known and unknown segments of a document. Instead of combining different indicators identifying the type of a document, we introduce HMMs, one for each type of document considered, to identify the type of a new document based on the documents in a given corpus. As such, the HMMs form an ensemble that can be easily extended with HMMs for new document types or weighted to prioritize documents of a certain type. As the type detection is carried out using the most likely sequence in the HMMs, we are able to use this sequence to pinpoint locations in a document that may contain new data.

Future work includes transfer learning such that trained models can be easily adapted for different corpora. Additionally, we consider forming a probability distribution over the ensemble of HMMs, leading to a distribution over distributions allowing for soft decisions about the type of document. Currently, we focus on improving the performance of document type detection regarding the distinction between documents from d_{sim} and d_{rev}.

References

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

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

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

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

[5] 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.

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

[7] T. Reuters, "Opencalais," *Retrieved June*, vol. 16, 2008.

[8] 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.

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

[10] H. Cunningham, V. Tablan, A. Roberts, and K. Bontcheva, "Getting more out of biomedical documents with gate's full lifecycle open source text analytics," *PLoS Computational Biology*, vol. 9, no. 2, 2013.

[11] J. Brank, G. Leban, and M. Grobelnik, "Semantic annotation of documents based on wikipedia concepts," *Informatica*, vol. 42, no. 1, 2018.

[12] 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.

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

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

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

[16] 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.

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

[18] 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.

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

[20] 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.

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

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

[23] 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.

[24] 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.

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

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

[27] 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.

[28] 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.

[29] J. T. Tou and R. C. Gonzalez, *Pattern recognition principles*. United States, 1974.