Enhancing Attentive Task Search with Information Gain Trees and Failure Detection Strategies

Kristin Stamm[1,2] and Andreas Dengel[1,2]
[1] *German Reseach Center for Artificial Intelligence, Trippstadter Str. 122, Kaiserslautern, Germany*
[2] *Department of Computer Science, University Kaiserslautern, Kaiserslautern, Germany*

Keywords: Machine Learning, Enterprise Applications, Evidence based Search, Information Gain Tree, Search Failure Detection, Degree of Belief, Dempster Shafer Theory, Multichannel Document Analysis.

Abstract: Enterprises today are challenged by managing requests arriving through all communication channels. To support service employees in better and faster understanding incoming documents, we developed the approach of process-driven document analysis (DA). We introduced the structure Attentive Task (AT) to formalize information expectations toward an incoming document. To map the documents to the corresponding AT, we previously developed a novel search approach that uses DA results as evidences for prioritizing all AT. With this approach, we consider numerous task instances including their context instead of a few process classes. The application of AT search in enterprises raises two challenges: (1) Complex domains require a structured selection of well performing evidence types, (2) a failure detection method is needed for handling a substantial part of incoming documents that cannot be related to any AT. Here, we apply methods from machine learning to meet these requirements. We learn and apply information gain trees for structuring and optimizing evidence selection. We propose five strategies for detecting documents without ATs. We evaluate the suggested methods with two processes of a financial institution.

1 INTRODUCTION

Today, enterprises are truly challenged by the management of new communication channels, such as email, having to deal with information overload. According to Bellotti et al., the quantity and the complexity of incoming requests can explain this overload (Bellotti et al., 2005). Enterprises strive toward managing the multichannel complexity, but fail when relying on existing IT solutions. The systems often suffer from the fragmentation between communication channels and also from the lack of connection to internal processes leading to information gaps. Mostly, their functionality is limited to legal requirements only. Instead, users need a system that helps them understanding the request, finding the related process instance and extracting relevant information.

We proposed the approach of process-driven document analysis (DA) (Stamm and Dengel, 2012a). A document arriving through an input channel is mapped to the corresponding process instance or task. The information expectations of the process instance are then used for conducting an analysis of the document and for extracting all relevant information. We applied two concepts: (1) *Attentive Tasks (ATs)* formally describing information expectations toward incoming documents and (2) the *Specialist Board (SB)*, first introduced by (Dengel and Hinkelmann, 1996), describing all available DA methods. First, we generate a DA plan. Second, we extract information about the document according to the DA plan. Finally, we search the corresponding AT based on this information. If necessary, the DA plan is adopted according to the AT. All steps are repeated until the matching AT is found and no more information can be extracted.

In this paper, we focus on improving the novel AT search algorithm that enables mapping on a task instance level. In (Stamm and Dengel, 2012b), we use DA results as evidences for prioritizing the available AT set by calculating a degree of belief (DoB) for each AT and evidence before combining them with Dempster-Shafer theory (Shafer, 1976). First evaluations demonstrated promising search results and good robustness, but also that the selection of initial evidences is crucial to search performance. Applying our approach to enterprises raises two challenges: (1) We believe that the introduced two step evidence structure remains insufficient for domains with more evidence

Stamm K. and Dengel A..
Enhancing Attentive Task Search with Information Gain Trees and Failure Detection Strategies.
DOI: 10.5220/0004227300810090
In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence* (ICAART-2013), pages 81-90
ISBN: 978-989-8565-39-6

types. Learning a more sophisticated structure is necessary. (2) The approach fails when no AT fits the document which is either the case when a new process is triggered by an incoming document, or when the mechanism of generating ATs failed due to processing errors. A failure detection method is needed.

Our goal is to address these enterprise requirements by enhancing AT search with methods from machine learning: (I) Learning evidence type trees by maximizing the average information gain, (II) using strategies for detecting search failure: (i) identifying evidences that appear only in documents leading to a new AT, (ii) comparing the maximum DoB to an expected DoB, (iii) and introducing general AT templates, (iv) as well as hybrid combinations from the previous strategies. We evaluate all approaches on a corpus from a financial institution.

Next, we review related work. We give a brief overview on our approach of process-driven DA focusing on the concept of ATs and the search algorithm. We then introduce our two approaches and present the results of their evaluations. Finally, we draw conclusions and give an outlook on future work.

2 RELATED WORK

There exist many approaches for mapping documents to processes or tasks - especially in the email environment, but they all have numerous drawbacks making them insufficient to our problem. They consider usually a few processes and ignore important process context information instead of numerous task instances. They are often costly to transfer to new domains, and they do not respect the importance of search criteria or search failure.

Some approaches rely on heuristics for mapping documents to tasks, e.g., *thrasks* that are a combination of conversation threads and tasks (Bellotti et al., 2005) or other context information for aggregation (Katz and Berman, 2011). They assume a direct connection between heuristic and task. Other approaches use established classification methods, like Naïve Bayes or Support Vector Machines. For example, Cohen et al. classify emails into sender intentions based on verb-noun pairs called *speech acts* (Cohen et al., 2004) as well as Scerri et al. who apply rule based task classification with speech acts (Scerri et al., 2010). Dredze et al. combine classification methods that rely on involved people or topics (Dredze et al., 2006). Faulring et al. propose a regular logistic regression algorithm for task type classification (Faulring et al., 2010), whereas Granitzer et al. pursue to aggregate tasks from user interaction behavior (Granitzer et al., 2009). Unfortunately, all these approaches do not involve a dynamic task set and are, therefore, not applicable to Attentive Tasks (ATs). Krämer recognizes the importance of tasks instances but uses manual task assignment (Krämer, 2010). Only Kushmerick and Lau use unsupervised learning for deriving process structures from emails. Their approach is applicable to personal email management with unstructured and implicit processes (Kushmerick and Lau, 2005). However, their approach has a limited applicability for well-defined processes as they appear in enterprises. Overall, none of the existing approaches consider search criteria, as for example, results from DA, or provide handling of search failure.

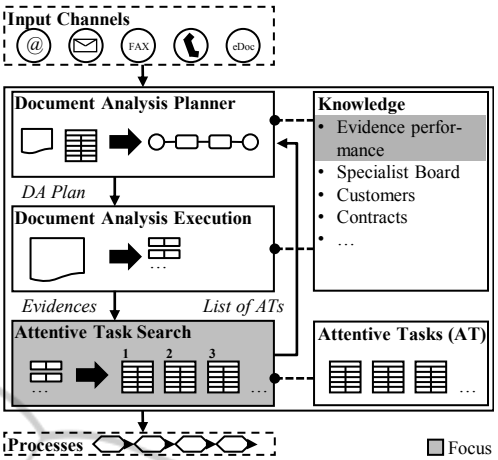

Figure 1: Process-driven document analysis system.

3 OVERALL APPROACH

This section presents the process-driven document analysis (DA) approach focusing on Attentive Task (AT) search (Stamm and Dengel, 2012a; Stamm and Dengel, 2012b). We detail concept, AT terminology and generation. We conclude with the main challenges from enterprise application.

3.1 Process-driven Document Analysis

The basic elements of the process-driven DA system are depicted in Figure 1. The system deals with documents coming from the main input channels in enterprises: email, mail, fax, call center, and eDocs. The system's core consists of the DA planner, the DA executor, and an AT search module. The system iteratively analyzes the document according to a plan and searches for the corresponding AT in the available

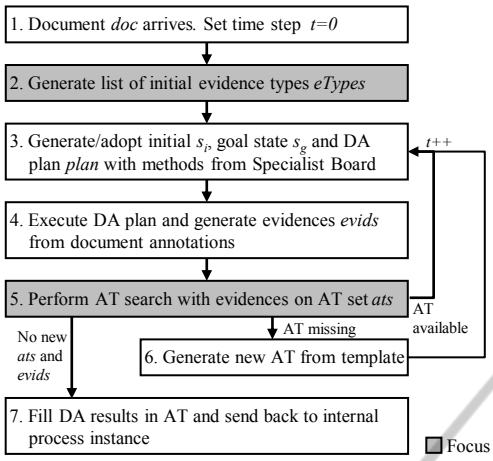

Figure 2: Main steps of the process-driven document analysis algorithm.

AT set. The modules use knowledge about evidence types, available DA methods described in the Specialist Board (SB), and enterprise knowledge. This work focuses on AT search and the use of initial evidences.

The main steps of the algorithm are outlined in Figure 2. When a document arrives, the system needs to decide which evidences have to be extracted initially. Based on these evidence types, it generates the initial state, the current goal state, and a DA plan. This plan is then executed and DA results are created in form of annotations. An annotation contains at least a type, value, and reference to the text sequence. Annotations are used as evidences for performing priority search on the set of available ATs. If there is a fitting AT and more information needs to be extracted, the system adopts the goal state, the initial state, and the DA plan before proceeding. If the AT is detected as missing, the corresponding AT template is identified for generating a new AT and adopting the goal. The algorithm stops when no more information is necessary or available.

3.2 Attentive Tasks

In the following, we detail the formalism of the AT, the AT generation options, and the AT search algorithm, as well as the challenges in enterprises.

3.2.1 Terminology

An *Attentive Task (AT)* describes a process instance's information expectations toward an incoming document at one step. The information expectations are represented by a list of *slots*, where each slot contains a *descriptor*, a *value*, and an *information type*, as well as *constraints* – in case the value is not available.

Table 1: Example of an Attentive Task.

Descriptor	Value	Type	Constraints
SenderEmail	anna@blue.org	EmailAddress	in(customer.email)
SenderName	Anna Blue	Person	in(customer.name)
RequestClass	ChangeOfOwner	Class	in(requestClasses)
NewOwnerName	Klaus Mustermann	Person	-
NewOwnerDoB	?	Date	DD.MM.YYYY
AdmissionOffice	?	Organization	in(organizations)

?: New value expected

Table 1 depicts an example of an AT for a change of contract owner request. It contains information known from previous steps of the process, e.g., customer name. Additionally, there is new information expected, e.g., about the owner's date of birth.

An *AT template* contains no process instance information and is used for generating ATs.

3.2.2 Generating Attentive Tasks

Since ATs formalize the information expectations of internal processes, they need to be generated by them. For example, a service employee processes a customer request, sends out a request, asking for missing information, and waits for reply. At this point, an AT is generated, i.e., the correct AT template is filled with known and expected information of the process instance. The AT is added to the pool of active ATs.

Depending on the enterprise's IT infrastructure, there are several options for triggering AT generation:

Manual Generation. The user decides himself to generate a new AT. He manually selects the template, fills in all information, and stores the AT on the server. This option requires high manual effort and might result in a high error rate.

Full Automatic Generation. The ideal generation of an AT is driven by an underlying system. This can be a workflow, an ERP[1], or any other system. Depending on the size and type of the system (e.g., standard software or self-tailored solutions) this option requires expensive customization. It is recommended to use existing APIs[2] of these systems for keeping adaptation effort low.

Supported Generation. An AT generation software supports the user in generating new ATs. It operates on the user's computer and communicates with a central system. Whenever the user needs to generate a new AT, he uses the system with a few selections. Due to system independence, this

[1]ERP = Enterprise Resource Planning
[2]API = Application Interface

option is less cost intense but also less convenient than the full automatic generation.

Depending on each process and its underlying systems, the generation method can be a mixture of them.

A complete set of ATs requires full process knowledge and perfect generation of ATs either by employees or by the systems. Each of these factors can fail resulting in an incomplete set of ATs.

3.2.3 Attentive Task Search

In our previous work, we proposed and evaluated Algorithm 1. It performs the prioritization of a set of ATs using DA results as evidences. Each evidence contains a descriptor d and a value v. The algorithm has an expected runtime of $O(n^2)$.

Algorithm 1: Search Attentive Tasks *atList* given evidences *evidList* (Stamm and Dengel, 2012b).

```
function ATSEARCH(atList,evidList)
    for all e in evidList do
        for all a in atList do
            d ← a.containsDescr(e.descr)
            v ← a.valMatch(e.descr,e.value)
            vs ← a.valueSetMatch(e.descr,e.value)
            p_e(a) ← mass(d,v,vs)
        end for
        m_e ← normalize(p_e)
        m_all ← combine(m_all,m_e)
    end for
return heapsort(atList,m_all)
end function
```

For each evidence and AT, a degree of belief (DoB) is calculated by assigning a mass value and normalization over the search set. All normalized mass functions m_e are combined with the Dempster Shafer rule (Shafer, 1976). See more details in (Stamm and Dengel, 2012a).

Evaluations showed that the AT search is robust in terms of parameterization. We also examined that selected evidence types influence search performance and that an evidence structure is needed for optimizing search performance. A two level structure was sufficient for our simple evaluation corpus. We believe that for larger evidence type sets, as appearing in enterprises, we need a more sophisticated structure.

3.2.4 Enterprise Requirements

The application of AT search faces two main challenges in the enterprise environment:

Initial Evidence Structure. We need a structure for the initial evidence-based search for prioritizing the AT set with a minimum number of search steps, i.e., number of evidences used for search.

Identification of Documents without Attentive Tasks. We need a failure detection strategy to identify documents without a corresponding AT - either documents that trigger a new process or cases where AT generation failed. So we can avoid processing errors.

4 INFORMATION GAIN TREES

In this section, we present a structure for deciding initial evidence extraction for search. This structure is build with supervised learning and applied automatically to Attentive Task (AT) search. We propose to generate evidence trees labeled with the average information gain at this level. In the following, we introduce the tree structure, the supervised learning algorithm and the integration into the AT search.

4.1 Learning

Figure 3 displays an exemplary information gain tree. Apart from the root node, it contains nodes, each labeled with an evidence type and the average information gain value of this evidence type at the current level. Leaves are reached when no evidence type generates an information gain above a defined threshold. The tree span is limited by the number of evidence branches at each level.

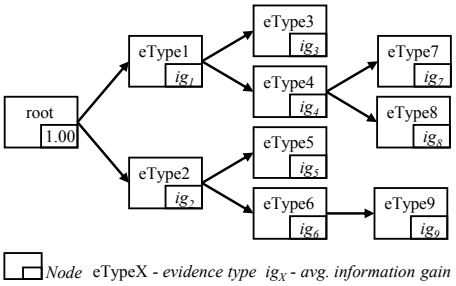

Figure 3: Information gain tree.

We rely on the concept of information gain IG introduced by Kullback and Leibler, because it measures the difference between the current entropy H and the expected entropy H_e after applying one more attribute (= evidence) e to the current search set (Kullback and Leibler, 1951). The information gain for an evidence e on a set of ATs A is defined as follows:

$$IG(A,e) = H(A) - H_e(A|e) \quad (1)$$
$$= \sum_{v \in val(e)} \frac{|\{a|a_e = v\}|}{|A|} H_e(\{a|a_e = v\})$$

where $a \in A$ and a_e is the slot in a with evidence type of e. Since the information gain depends on the current AT set, we learn average information gains for random AT sets repeatedly for each tree level.

Algorithm 2: Generate information gain tree.

```
function GENERATETREE(set,ets,tree,n)
    nodeList ← avgInfoGains(set,ets,tree,n)
    topList ← getTopItems(nodeList,thresh,branch)
    if isEmpty(topList) then return tree
    end if
    for all n2 in topList do
        tree ← addNode(tree,n,n2)
        tree ← generateTree(docs,ats,ets,tree,n2)
    end for
return tree
end function
function AVGINFOGAINS(set,ets,tree,n)
    usedEts ← getPath(tree,n)
    leftEts ← reduceList(ets,usedEts)
    for i ← 0 : i ≤ iterations; i++ do
        ats ← createRandomATSet(set)
        doc ← selectRandomDoc(ats,set)
        doc.evids ← analyseDoc(usedEts)
        prioAts ← atSearch(ats,doc.evids)
        subAts ← createSubgroup(prioAts)
        for all e in lefEts do
            g ← calcInfoGain(subAts,doc,e)
            infoGains(e).add(g)
        end for
    end for
    avgIGs ← calcAvgInfoGains(infoGains)
    nodes ← createNodes(leftEts,avgIGs)
return sort(nodes)
end function
```

The tree learning algorithm is outlined in Algorithm 2. It consists of two functions, *generateTree* and *avgInfoGains*. The first one is a recursive function that learns a tree *tree* for a given test set *set* consisting of documents and their related ATs, based on average information gain values. First, we calculate for all evidence types *ets* the average information gains and return a list of nodes *nodeList*. From this list, we select the top items limited by a branching factor *branch* and a fixed minimum threshold *thresh* for the information gain value. If the *topList* does not contain any more evidence types, we have reached a leaf and return. Otherwise, we add each node from *topList* to the tree and perform *generateTree* on the next level from the current node $n2$.

The function *avgInfoGains* calculates repeatedly information gains for the remaining evidence types *ets* on a random AT set *set*. First, we get a list of all used evidence types *usedEts* from the tree and reduce the list to the available evidence types *leftEts*. We repeat information gain calculation *iterations* times. For each iteration, we select a random AT search set *ats* and a corresponding document *doc*. Then, we extract all evidences *doc.evids* according to the used evidence types. AT search is performed. Based on the priority list of ATs, we select the remaining, matching AT subgroup *subAts*. For this subgroup, we calculate the information gain *g* for each evidence type. Finally, we calculate the average information gain values for each evidence type and return a sorted node list.

4.2 Application to Search

Algorithm 3 outlines, how we apply the learned information gain tree to AT search.

Algorithm 3: Extract evidences from document *doc* according to evidence decision tree *tree*.

```
function APPLYEVIDENCETREE(doc,ats,tree)
    while tree.hasNext() do
        node ← tree.getNextWithMaxInfoGain()
        e ← analyse(doc,node.eType)
        if e! = null then
            evidList.add(e)
        else
            tree.stepBack()
            tree.prunePaths(node.eType)
        end if
    end while
    prioList ← ATsearch(ats,evidList)
return prioList
end function
```

The function *applyEvidenceTree* generates one path of evidences *evidList* within the tree *tree* that can be extracted from the document *doc*.The DA results are applied as evidences to AT search. For generating the evidence list *evidList*, the next node *node* in the tree is selected by the maximum information gain assigned to the descendants of the current node. If this evidence type can be extracted from the document, the evidence *e* is added to the list of evidences. If not, we move one level up in the tree and prune all paths in the tree that include this evidence type. The resulting evidence list is used for AT search and the function finally returns a sorted AT list.

5 FAILURE DETECTION STRATEGIES

Sometimes, when a document arrives , there does not exist a matching Attentive Task (AT). This is either

the case if the document invokes a new process instance in the enterprise or if the generation of the AT failed (see Section 3.2.2). During AT search, we need to decide fast if there is an AT or not for avoiding processing errors. These search failures can be handled by creating a new AT from the corresponding AT template. In the following, we present three strategies and combinations of them for identifying such documents during or in advance to AT search.

5.1 Specific Evidence Types

Documents triggering a new process instance often contain evidence types that are not contained in other documents and vice-versa. These are most likely basic information that is not mentioned again during a conversation. We propose learning of evidence types specific for new requests and extracting them before AT search. Equation 2 formalizes the rule for determining AT failure for a document d depending on evidence types E_{new} specific for *new* documents:

$$fail_{spec}(d, E_{new}) = \begin{cases} 1.0 & \text{if } \exists e \in d.E | e.t \in E_{new} \\ 0.0 & \text{else} \end{cases} \quad (2)$$

where failure is true (1.0) if there exists at least one evidence e extractable from the document $d.E$, whose evidence type $e.t$ is contained in E_{new}. This approach is not costly in terms of search steps, because it does only require extraction steps. It cannot detect AT generation failures.

5.2 Expected Degree of Belief

The degree of belief (DoB) measures to which extent each AT matches the evidences from a document in comparison to all other ATs. Therefore, we propose to use the DoB value for detecting documents, where we believe that no AT matches. We compare an expected DoB dob_e for a set of evidence types E to the actual DoB of the first AT in the prioritized list dob_{top}. If the difference is beyond a threshold t, a failure is detected. Equation 3 formalizes failure detection for a document d, the AT set A, and an expected DoB function dob_e, as well as a threshold function t:

$$fail_{DoB}(d, A, dob_e, t) = \quad (3)$$
$$\begin{cases} 1.0 & \text{if } dob_e(d.E, |A|) - dob_{top}(d.E, A) > t \\ 0.0 & \text{else} \end{cases}$$

where failure is detected if the difference between the expected DoB dob_e and the actual top DoB after search dob_{top} is greater than the defined threshold.

This strategy postulates that the DoB of documents with existing AT differs significantly from documents without AT. We conduct evaluations in advance to confirm this assumption. Since the detection strategy includes information about the current AT set, it addresses both failure cases.

5.3 Attentive Task Templates

For failure handling we generate new ATs from AT templates. We, therefore, propose to add all AT templates to the AT search set. Equation 4 formalizes template failure detection for a document d, the current ATs A, and all templates T:

$$fail_{tem}(d, A, T) = \begin{cases} 1.0 & \text{if } a_{top}(d, (A \cup T)) \in T \\ 0.0 & \text{else} \end{cases} \quad (4)$$

where failure is detected if the top AT a_{top} after search on the combined AT set A and T is a template.
This strategy addresses both failure cases and is simple to implement, since it does not require learning.

5.4 Hybrid Strategies

We consider combining the different strategies for achieving better and faster detection results. The *Expected Degree of Belief* and *AT Templates* strategies exclude each other, because inserting templates in the search set prevents using the DoB of a non-matching AT on top of the list. Thus, we combine the *Specific Evidence Types* strategy with the other ones. First, we detect if a document contains an evidence type specific for new documents. If the document passes the test, we apply the second or third strategy. In this way, we expect to reduce the search steps for new documents and to improve overall detection success.

6 EVALUATION OF INFORMATION GAIN TREES

We conduct evaluations with the information gain tree structure. First, we learn the tree and apply it then to Attentive Task (AT) search.

6.1 Evaluation Setup

We perform the evaluations on a corpus generated from two business processes of a financial institution. The corpus includes 49 emails from probands that conducted requests toward a bank. Each document in the corpus has been annotated with document

Table 2: Information gain tree properties for alternating branches and thresholds.

	Number of nodes					Average depth				
	Threshold									
Branches	0.4	0.3	0.2	0.15	0.1	0.4	0.3	0.2	0.15	0.1
1	0	2	5	3	3	0.0	2.0	5.0	3	3
2	0	6	69	104	87	0.0	2.0	5.8	7	7,1
3	0	12	365	1,898	5K	0.0	2.0	5.9	7.8	8.9
4	0	15	792	9K	83K	0.0	2.0	5.8	7.9	10.1
5	0	18	1,392	23K	>1,000K	0.0	2.0	5.7	7.9	11.7

analysis (DA) results and we have generated an AT for each document. Our approach is general to all input channels, but we focus on email here for reducing complexity not relevant to search performance. It is possible to extend the approach to the other channels.

The evaluations of the information gain tree have been conducted in two steps:

1. **Tree Learning.** The proposed tree learning algorithm depends on the information gain threshold and the number of branches per node. We varied both parameters for analyzing resulting trees in terms of number of nodes and average tree depth.
2. **Tree Search.** The goal of using information gain trees is to provide a structure that reaches good search results in a minimal number of search steps. We compare the tree setups generated during learning and compare them to simpler structures as random selection from all evidences (*All*), a set of best performing structures (*Top7*), and the two level structure (*Top7 2-Level*).

6.2 Tree Learning

The overall goals of implementing information gain trees are the minimization of search and extraction steps, while keeping good search results, creating a structure that is robust to non-extractable evidences, and minimizing learning time.

During learning evaluation, we, alternated the two main parameters - branch factor and threshold - to terminate branch extension. For each tree, we counted the number of nodes that directly correlates to learning time, and the average tree depth that influences number of search steps. We derive the main findings depicted in Table 2 as follows:

Learning Time. The lower the threshold and higher the branch factor is, the larger the tree becomes – growing exponentially. In terms of learning time, the threshold should be limited to 0.2, whereas branches to 4. A threshold of 0.4 or higher does not generate any node (except the root node).

Search Steps. Average tree depth depends on the threshold and increases when decreasing the threshold. Experienced in simple search, good search results derive from 3 or more search steps. A threshold between 0.3 and 0.4 is corresponding.

Branches. The more branches, the more robustness to not extractable evidences we achieve. Therefore, a high branching factor is preferable, but does also influence learning time tremendously. The results show, that a limitation of the number of branches, can help to limit overall tree size.

We conclude that the limitation of threshold and branches is necessary for limiting tree learning time.

Table 3: Information gain tree search results.

	AvgRank				Search steps (extraction)			
	Threshold							
Branches	0.3	0.2	0.15	0.1	0.3	0.2	0.15	0.1
1	0.74	0.75	0.74	0.74	1.1 (2.0)	1.3 (2.2)	1.4 (2.2)	1.3 (2.2)
2	0.75	0.74	0.75	0.75	1.1 (2.9)	1.3 (3.1)	1.3 (3.1)	1.3 (3.1)
3	0.75	0.74	0.75	0.74	1.1 (3.7)	1.3 (4.0)	1.3 (4.0)	1.3 (4.0)
4	0.75	0.74	0.74	0.74	1.2 (4.6)	1.4 (5.0)	1.4 (5.0)	1.4 (5.0)

Table 4: Performance of previous methods.

Method	Evidences	AvgRank	Search steps (extraction)
All	E=6	1.04	6.0 (6.0)
	E=7	0.77	7.0 (7.0)
Top7	E=3	1.17	3.0 (3.0)
	E=4	0.54	4.0 (4.0)
Top7 2-level	E=2	0.81	2.0 (2.0)
	E=3	0.59	3.0 (3.0)

6.3 Tree Search

For tree search, we compare search performance and runtime optimization for each tree configuration. Then, we compare tree search with the three previously used methods for determining initial evidences: *All*, *Top7*, and *Top7 2-level*. For search performance, we use the average ranks as main measure. Optimization is measured in search and extraction steps.

We summarize the results in Table 3 as follows:

Average Rank. All trees perform similar with an average rank between 0.74 and 0.75. A small, simple tree structure is sufficient for achieving good search results for our corpus. The evidences in the first branch are most likely extractable.

Search Steps. Average number of search steps is low (from 1.1 to 1.4) and increases slightly with decreasing threshold. This supports that in most cases the first branches are used for search.

Extraction Steps. Average number of extraction steps increases from 2.0 up to 5.0 with decreasing threshold and increasing branching factor. When

an evidence in the first branches cannot be used, several extraction steps are necessary.

Comparing the tree search results to the previous methods (see Table 4) reveals that tree search reaches similar search performance as *All* between 6 and 7 evidences, as *Top7* between 3 and 4, and as *Top7 2-level* between 2 and 3. We infer that the main advantage of tree search is the optimization of search and extraction steps in comparison to simpler methods.

In conclusion, we found a structure that optimizes search and extraction steps and delivers good search results with relatively low effort in training time and calibration of the method. We believe, the information gain trees will enable our system to deal with more complex setups.

7 EVALUATION OF FAILURE DETECTION

In this section, we evaluate the different failure detection strategies regarding detection performance. We pre-evaluate the degree of belief (DoB) in dependence on the search set size and the evidence type to prepare the second strategy (expected DoB).

7.1 Evaluation Setup

We evaluate on the same corpus in two steps:

1. **Degree of Belief Values.** We conduct Attentive Task (AT) search on random search setups for evaluating the dependency of the degree of belief (DoB) value on search set size and evidence type. Further, we repeat the experiment for understanding, how DoB develops in case of a search failure, and for generating a threshold. We repeat each search setup 20,000 times.
2. **Failure Detection Strategies.** We evaluate each of the proposed strategies: 1) specific evidence types, 2) expected DoB, 3) inclusion of AT templates, as well as hybrid strategies, 1) & 2) and 1) & 3). We compare them with established classification measures: precision $Pr = tp/(tp+fp)$, recall $Re = tp/(tp+fp)$, and accuracy $Acc = (tp+fp)/(tp+fp+tn+fn)$. True positives tp are correctly detected failures, false positives fp non-failures classified as failures, true negatives tn correctly detected non-failures, and false negatives fn not detected failures. We also conduct a separate evaluation of the two failure cases. Additionally, we aim at minimizing the costly search steps. Experiments were repeated 80.000 times for each setup including varying number of search steps from 1 to 6, which are the number of evidences used for search.

7.2 Degree of Belief Evaluation

We repeated DoB experiments for different ATs search sets and varied the search set size, number of evidences as input for search, and the type of evidence group. For evidence groups, we differentiate between the best performing evidence types from our previous work (*Top7*), all evidence types (*All*), and all possible evidence types without the *Top7* (*All w/o Top7*). For each search experiment, we generate a random AT set of random size, select one corresponding document, extract evidences according to the evidence type group, and execute search. The findings depicted in Figures 4 (a) - (c) are summarized as follows:

Search Set Size. The larger the search set is, the smaller the DoB value of the corresponding AT. Figure 4 (a) displays the DoB development for all evidence types when one evidence is used. This effect is diminished with increasing number of evidences (see Figure 4 (b)). To measure the development over search set size we calculate the compound growth rate (CGR)[3] between search set size 2 and 17 for different evidence numbers.

Evidence Type Performance. The well performing evidence types (*Top7*) have less decreasing influence on the DoB value than the others. We derive that DoB is more stable for calibrated searches.

Search Failures. Comparing the DoB for successful searches and the top DoB for failures shows that only a few *selected* evidence type combinations result in relevant differences between the values (see Figure 4 (c)). This is caused by many evidences also matching to one or more incorrect ATs. In such a case, the ATs get a higher matching value and after normalization they get a value similar to the correct AT. For the expected DoB strategy, we use only the selected evidence types and half of the average difference as threshold.

We conclude that in our setup the DoB value is highly sensitive to search set size and evidence type. There are only a few evidence type combinations that allow to use the DoB distance to an expected DoB for identifying if the corresponding AT is not included in the search set. We will further evaluate the related strategy, but these results indicate that the expected DoB strategy could become fragile in other domains.

[3] $CGR(s_0, s_n) = (\frac{avgDoB(s_n)}{avgDoB(s_0)})^{\frac{1}{s_n - s_0}} - 1$, s_i is the search set size and $avgDoB(s_i)$ the average DoB for search set size s_i

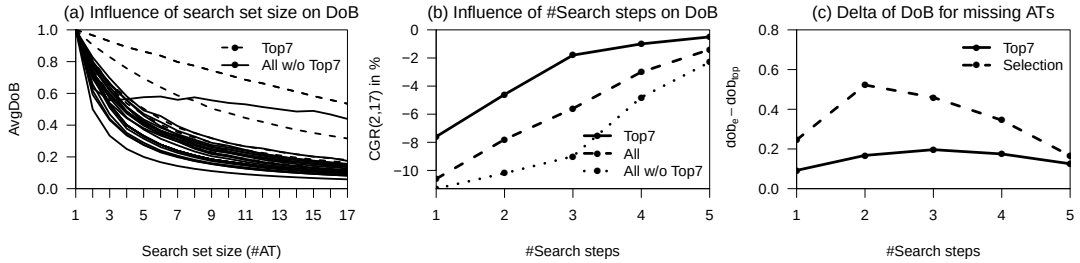

Figure 4: Influencing factors on Degree of Belief (DoB) values: (a) Influence of search set size for all evidence types on DoB, (b) Influence of search steps (= no. of evidences) on average compound growth rate of DoB, (c) and influence of search steps and selection of evidence types on distance between expected DoB and top DoB when the Attentive Task is missing.

Table 5: Classification performance of the failure detection strategies.

	1) Evidence types			2) Expected DoB			3) Templates			1) & 2)			1) & 3)		
#Search steps	**Pr**	**Re**	**Acc**	**Pr**	**Re**	**Acc**	**Pr**	**Re**	**Acc**	**Pr**	**Re**	**Acc**	**Pr**	**Re**	**Acc**
0	1.00	0.23	0.47	-	-	-	-	-	-	1.00	0.23	0.47	1.00	0.23	0.47
1	-	-	-	0.77	0.54	0.57	1.00	0.23	0.34	0.82	0.71	0.69	1.00	0.52	0.59
2	-	-	-	0.93	0.80	0.82	0.99	0.53	0.60	0.94	0.87	0.87	0.99	0.73	0.77
3	-	-	-	0.93	0.77	0.80	0.99	0.73	0.76	0.94	0.86	0.86	0.99	0.86	0.87
4	-	-	-	0.93	0.59	0.69	0.98	0.85	0.86	0.94	0.75	0.80	0.98	0.93	0.93
5	-	-	-	0.92	0.51	0.63	0.98	0.88	0.88	0.94	0.70	0.76	0.98	0.94	0.94
6	-	-	-	0.92	0.39	0.55	0.98	0.91	0.90	0.95	0.62	0.71	0.98	0.96	0.95

7.3 Failure Detection Strategies

We evaluate the five proposed detection strategies on a randomly generated AT set for a randomly selected document. We alternate the number of evidences used for search, because this is the most expensive calculation step ($O(n^2)$). We repeat all experiments for search failure and for non-failure.

For documents triggering a new process compared to documents related to one process instance, we use the ratio from the corpus (new: 38%, instance: 62%). For the case of AT generation failure, we assume a 50% ratio. We expect a much lower ratio in enterprise application. Due to the dependency on the AT generation approach, it is difficult to predict this ratio. We apply the same ratios for all strategies, so the results remain comparable.

The main findings are depicted in Table 5 and a separate overview of the two cases of search failure is displayed in Figure 5 (a) and (b). We summarize the results for each strategy as follows:

1) Selected Evidences. This strategy does not involve any search steps. Precision is optimal (1.0) whereas recall (0.23) and accuracy (0.47) are very low. The reason for the discrepancy is that the strategy only detects documents triggering a new process (see Figure 5 (a) & (b)).

2) Expected DoB. This strategy performs optimal when using two evidences. It reaches precision of 0.93, recall of 0.80, and accuracy of 0.82. According to the DoB pre-evaluations, the difference between expected DoB and DoB in case of failure is best differentiating and leads to good results. The development is for both failure cases similar.

3) Templates. When including AT templates, precision decreases slightly (from 1.00 to 0.98) with increasing search steps, whereas recall and accuracy increase tremendously (from 0.23 to 0.91, from 0.34 to 0.90). This correlates to the general AT search development, where increasing number of evidences improve search results. The strategy performs better for generation failures than new documents. A similar performance to strategy 2) at search step two is reached with four.

1) & 2). The combination leads to better failure detection performance overall. There is again an optimum for two search steps. Precision reaches 0.94, recall 0.87, and accuracy 0.87. This is caused by the improvements in new document detection (see Figure 5 (a)).

1) & 3). This hybrid strategy also improves detection performance. Accuracy (from 0.47 to 0.98) and recall (from 0.23 to 0.96) increase with number of search steps. Precision decreases again from 1.0 to 0.98. The improvements are caused by improvements in new email detection (see Figure 5 (a)). A similar performance to strategy 1) & 2) at search step two is reached with three.

We conclude that it is recommendable using a hybrid strategy. The combination with the expected

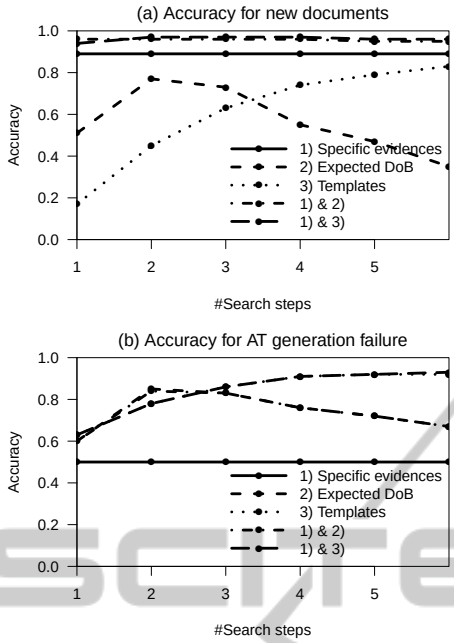

Figure 5: Accuracy of the failure detection strategies separated by the case of absent AT: (a) New document without related AT, (b) AT of the document is missing.

DoB strategy results in best detection performance with only two search steps. Nevertheless, this strategy also has drawbacks. The learning of evidence combinations is intricate, the results seem fragile and depend on the application domain. Additionally, the required evidences for a good detection performance might not perfectly fit with the optimal evidences for initial search. Hence, we hesitate recommending this strategy in general for any domain.

The combination with templates appears more reliable, even if it produces similar results as 1) & 2) with three search steps. The selection of evidences for search is compatible with initial evidence selection and the method does not require pre-learning. We, therefore, recommend implementing template search in the first place and evaluating the performance of expected DoB in the particular domain.

8 CONCLUSIONS

In this paper, we propose two approaches enhancing our existing AT search algorithm and making it more robust to real world requirements: (i) information evidence gain trees and (ii) missing Attentive Task (AT) detection strategies. First evaluations already show that the application of trees significantly minimizes search steps and we expect it to support even more complex search domains. For search failure detection strategies, we found that combining the specific evidence type decisions with the two strategies integrating the search set performs best. For simplification, we recommend applying first the insertion of AT templates and then investigating whether the expected degree of belief (DoB) comparison is applicable to the corresponding domain or not.

For future work, we plan transferring the results to more complex domains and increasing the search set size. Further, we aim at evaluating DA planning.

REFERENCES

Bellotti, V. et al. (2005). Quality vs. quantity: Email-centric task-management and its relationship with overload. *Human-Computer Interaction*, 20:1–2.

Cohen, W., Carvalho, V., and Mitchell, T. (2004). Learning to classify email into ”speech acts”. In *Proceedings of EMNLP*.

Dengel, A. and Hinkelmann, K. (1996). The specialist board - a technology workbench for document analysis and understanding. In *IDPT*.

Dredze, M., Lau, T., and Kushmerick, N. (2006). Automatically classifying emails into activities. In *Proceedings of IUI*.

Faulring, A. et al. (2010). Agent-assisted task management that reduces email overload. In *Proceedings of IUI*.

Granitzer, M. et al. (2009). Machine learning based work task classification. *Journal of Digital Information Management*.

Katz, A. and Berman, I. (2011). Designing an e-mail prototype to enhance effective communication and task management: A case study. *Serdica Journal of Computing*, 5.

Krämer, J. (2010). Pim-mail: consolidating task and email management. In *Proceedings of CHI*.

Kullback, S. and Leibler, R. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22.

Kushmerick, N. and Lau, T. A. (2005). Automated email activity management: an unsupervised learning approach. *Proceedings of IUI*.

Scerri, S., Gossen, G., Davis, B., and Handschuh, S. (2010). Classifying action items for semantic email. In *Proceedings of LREC*.

Shafer, G. (1976). *A Mathematical Theory of Evidence*, volume 1. Princeton university press Princeton.

Stamm, K. and Dengel, A. (2012a). Attentive tasks: Process-driven document analysis for multichannel documents. *Proceedings of DAS*.

Stamm, K. and Dengel, A. (2012b). Searching attentive tasks with document analysis evidences and dempster-shafer theory. *Proceedings of ICPR*.