Automatic Task-Cluster Generation based on Document Switching and Revisitation

Charlie Abela[1], Chris Staff[1], and Siegfried Handschuh[2]

[1] Department of Intelligent Computer Systems,
University of Malta, Malta
{charlie.abela,chris.staff}@um.edu.mt
[2] Department of Computer Science and Mathematics,
University of Passau, Bavaria, Germany
{siegfried.handschuh}@deri.org

Abstract. Personal Information Management (PIM) research is challenging primarily due to the inherent nature of PIM. Studies have shown that people often adopt their own schemes when organising their personal collections, possibly because PIM tool-support is still lacking. In this paper we investigate the problem of automatic organisation of personal information into task-clusters by transparently exploiting the user's behaviour while performing some tasks. We conduct a controlled experiment, with 22 participants, using three different task-execution strategies to gather clean data for our evaluation. We use our PiMx (PIM analytix) framework to analyse this data and understand better the issues associated with this problem. Based on this analysis, we then present the incremental density-based clustering algorithm, iDeTaCt, that is able to transparently generate task-clusters by exploiting document switching and revisitation. We evaluate the algorithm's performance using the collected datasets. The results obtained are very encouraging and merit further investigation.

Key words: Density-Based Clustering, Personal Information Management, Task Clusters

1 Introduction

When we are performing some task on our desktop, we tend to spend a considerable amount of time looking back, establishing past references and remembering [10]. Whether performing the task requires us to search for information on the Web, reply to some email we've received, or resume writing some other document which we've worked on the day before, we tend to rely on our organisational skills and the support of search, bookmarking and history tools [9].

The common feature in these tools is their ability to help us find or re-find information by exploiting revisitation [10]. However, most of these tools tend to consider the user's information-seeking activities as unrelated events, unlike the way we actually organise things, which is usually in terms of directories (on our desktop) and tasks (conceptually) [11, 13]. Furthermore, humans are by nature or by force multi-taskers, and tool-support should cater for situations whereby users switch between one task and another or are interrupted [4, 11].

In this paper we align our research with efforts such as those of [12, 1] and investigate how to transparently cluster documents such as Web-browsed documents, office-related documents and emails, which are viewed by the user and belong to the same task.

As documents are presented to users in windows and tabs, and users switch between them, we collect and exploit evidence of the users *"window switching behaviour"* to identify and generate task-clusters[3]. These clusters can be used to re-find specific task-related documents and to resume a task, if or when, the user is interrupted. Currently, we do not label the identified task-clusters and we are not considering the content of the accessed documents as was the case in [12].

We adopt an unsupervised method that treats the accesses to documents as an undirected *activity graph*, $G_a(V, E)$, based on which we create the task-cluster's graph $G_c(V, E)$, which is both weighted and undirected. The edge weight $w(u, v)$ in $G_c(V, E)$ reflects the strength of the association between two nodes u and v. In the text we tend to use interchangeably the words "document" and "node", depending on the perspective we consider.

We performed a controlled experiment, conducted with 22 participants, using 3 different task-execution strategies to gather user's window-switching behaviour data for our evaluations. We separate the collected data into 3 groups depending on the task-execution strategy adopted: in succession with no interleaving (used as baseline), interleaved and interleaved over different sessions. We also developed the PIM analytix *PiMx* framework through which it was possible to simulate, off-line, the task-execution over the collected data and at the same time exploit network-analytics to analyse and visualise $G_a(V, E)$ and $G_c(V, E)$. Through PiMx we were able to understand better which algorithmic approach was more suitable.

In our approach we have factored-in an important feature, which to our knowledge has not been addressed yet. We refer to the incremental nature of the task and task-clusters, with nodes and edges being added over time as the user visits and re-visits documents. We extend the work of [6], and propose an incremental density-based clustering algorithm, which we call **iDeTaCt** that identifies the dense regions from the less denser ones in the accessed documents' space, identifying the task-clusters in the process.

We evaluated our approach to verify how well our algorithm is able to: (i) identify those nodes that belong together in a task-cluster and (ii) identify when a switch between two nodes is effectively a task-switch. The results are very encouraging and **iDeTaCt** managed to cleanly separate all the tasks, using the data of all the participants from the baseline group which performed the tasks in succession. When we used the data with interleaved tasks, **iDeTaCt** was

[3] A task-cluster is a group of documents that pertain to a task.

found to be sufficiently reliable in more than 50% of the cases, at the expense of capturing less documents from the referenced tasks.

The rest of the paper is structured as follows. In Sec. 2 we present research which is closely related to our own. We follow up with a detailed description of the controlled experiment that we conducted to collect reliable data. In Sec. 4 we give an overview of the PiMx framework which we use to analyse the collected data. We introduce our *iDeTaCt* algorithm in Sec. 5, which is followed by the evaluation and future work sections.

2 Related Work

In our work we draw parallels with research related to task-switching and identification, and others that have adopted the graph data structure as the underlying representation for the user's switching and revisitation behaviour.

In his thesis [10], Mayer presented an integrative history, visualization tool entitled SessionGraphs. The tool allows a user to view her browsing activity as an animated, interactive graph and to organise the visualisations according to her tasks. We have adopted a similar graphical approach for our PiMx framework, however the scope behind PiMx was that of allowing a researcher to better understand the issues related to the problem of automatic task-cluster generation rather than to provide browsing support.

The search bar presented in [11] persistently maintains a hierarchical Web history organised around search topics and queries. It assists users in organising complex searches and re-acquiring the context of a suspended search, based mainly on topic and query-driven groupings. The multitasking bar presented by [13] copes with both multiple tasks as well as multiple session tasks. They considered a task as having different states and it was up to the user to maintain and organise a task. Our approach aims to transparently automate the task-cluster generation without requiring any user intervention.

A semi-automated approach to task-identification was adopted by [3] which used activity-log analysis to group the accessed resources based on cues generated by an individual while performing some task. We adopt this same approach to collect the user's activity information through dedicated application plug-ins. We maintain a global desktop history with information about all the created, accessed and edited documents which also includes Web-related documents and queries.

To identify documents pertaining to a task [12] computed document content similarity and applied a maximal clique-finding algorithm over the user's switching activities. Unlike this approach, we currently do not intend to exploit the content of the accessed documents, however we will consider the incremental characteristics underlying an information-finding process.

In [1] a PageRank-like association heuristic was used to compute the association between windows opened on the desktop and presented a visualisation through which windows that were frequently clicked in sequence, were displayed closer together. However, no task-clusters were explicitly generated.

In [7, 8] an incremental density-based graph clustering approach is used to cluster documents and find interesting subgraphs, respectively. Density-based clustering is quite interesting since it is capable of coping effectively with noise. In our case this will be a major challenge since users tend to constantly switch between tasks, with the result that it would be more difficult to deal with those documents that are accessed in between tasks.

3 Controlled Data Collection Experiment

Evaluating PIM related research is inherently difficult, in particular due to the lack of readily available datasets. We therefore conducted a data-collection experiment in a controlled environment, to collect clean data related to the window-switching behaviour of the participants while performing some predefined tasks.

We set up a cluster of machines in one of our laboratories, each running Windows OS and having two activity-monitoring applications installed on them. One of the applications monitored browsing activity on Firefox[4] while the other monitored file browsing activity (e.g. of word processing documents) on the desktop. The participants were advised about this monitoring and were assured that the data would be anonymised and used only for the specified research purpose before the start of the experiment. This consisted of all the participants performing the same three, predefined information-seeking tasks, by answering a number of questions related to specific topics. Apart from seeking out information, participants had to compile a document with the relevant answers for each task, which they had to email to us at specified intervals. Our methodology was in line with that adopted by [11, 10].

The tasks required participants to provide specific information about the planning of a *vacation* in a specific country; answering questions related to the research area of *human computation*; and providing information about any two upcoming *music events*. The tasks were conducted either over single or multiple sessions. At pre-established intervals, unknown to the participants, we sent out emails that either informing them what a task entails or else requesting that they switch to another task.

There were in total 22 participants, 25% of whom were female. The participants were students and members of staff (lecturing and administration) from the Faculty of ICT within the University of Malta. The students were compensated €10 for their participation in the experiment.

The participants were split into three groups. Each group performed the experiment separately from the others. The groups were split as follows:

i. *Group 1*: the 7 participants in this group completed each of the three tasks in sequence, starting with task 1, followed by tasks 2 and 3, without interruptions. We use the data from this group as the baseline for our algorithm, since the tasks are clearly separated from each other;

[4] `https://www.mozilla.org/en-US/firefox/desktop/`

ii. *Group 2*: there were 10 participants in this group. They performed the three tasks in a single session. They started working on task 1 but were interrupted with an email from us requesting that they start task 2. After some more time we sent another email requesting the participants to stop working on task 2 and start working on task 3. We later interrupted them with yet another email requesting that they switch back to task 2, finish it, and then switch to, and complete tasks 1 and 3, in this order. In this way the tasks were interleaved and thus identifying which documents pertained to which task, becomes even more challenging;
iii. *Group 3*: the 5 participants in this group performed the tasks in the same order as Group 2 and with similar interruptions, however they were stopped 30 minutes into the session. Later on we asked them to continue the experiment in another session, which took place some days later. During the second session they had to resume the tasks and complete them in a sequential order with no further interruptions. In this way we wanted to introduce some more challenges to the participants, since they had to remember what they had been working on and recall the state of the task/s before they were stopped.

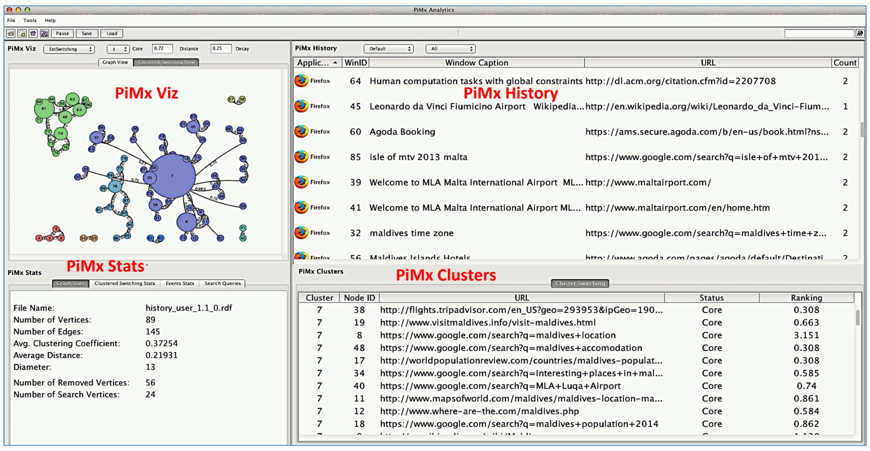
Although we tried to have an equal number of participants in each group, due to availability issues of our participants we had to somewhat relax this aspect. Furthermore, the data collected from one participant from Group 1 and another from Group 2 was unusable due to issues with the data-logging applications, which were unfortunately, not noticed in time.

The logged data included information about the type of event (e.g. navigational and tabbed events), the application that generated the event, the timestamp, the URL of the document accessed as a result of the event, an excerpt of text from the window caption. Other information, specific to particular events was also captured. This included, the URL of the page that was in focus before the event was triggered, as is the case of the navigational events. We also captured the file name and whether a document was edited or not, in the case of the desktop's file-related events. The data logged from each participant was anonymised and cleaned for further processing.

4 PiMx: tool for analysing the data

We implemented a tool, called PiMx (**P**ersonal **i**nformation **M**anagement analyti**x**) to analyse the collected data and understand better how we can algorithmically exploit document switching and revisitation to generate the task-clusters.

PiMx allows a researcher to load a user's activity-log and to simulate the execution of the task-trail for that user. This process can be paused and resumed at any time, allowing the researcher to analyse and compare the evolving task-trail through different views, see Fig. 1. The *PiMx-History* is similar to the tool developed by [5] and allowed us to view details related to all accessed documents, including the URI and amount of revisitations. It is also possible to filter the



File Name:	history_user_1.1_0.rdf
Number of Vertices:	89
Number of Edges:	145
Avg. Clustering Coefficient:	0.37254
Average Distance:	0.21931
Diameter:	13
Number of Removed Vertices:	56
Number of Search Vertices:	24

Applic...	WinID	Window Caption	URL	Count
Firefox	64	Human computation tasks with global constraints	http://dl.acm.org/citation.cfm?id=2207708	2
Firefox	45	Leonardo da Vinci Fiumicino Airport Wikipedia...	http://en.wikipedia.org/wiki/Leonardo_da_Vinci-Fium...	1
Firefox	60	Agoda Booking	https://ams.secure.agoda.com/b/en-us/book.html?ns...	2
Firefox	85	isle of mtv 2013 malta	https://www.google.com/search?q=isle+of+mtv+201...	2
Firefox	39	Welcome to MLA Malta International Airport ML...	http://www.maltairport.com/	2
Firefox	41	Welcome to MLA Malta International Airport ML...	http://www.maltairport.com/en/home.htm	2
Firefox	32	maldives time zone	https://www.google.com/search?q=maldives+time+z...	2
Firefox	56	Maldives Islands Hotels	http://www.agoda.com/pages/agoda/default/Destinati	2

Cluster	Node ID	URL	Status	Ranking
7	38	http://flights.tripadvisor.com/en_US?geo=293953&ipGeo=190...	Core	0.308
7	19	http://www.visitmaldives.info/visit-maldives.html	Core	0.663
7	8	https://www.google.com/search?q=maldives+location	Core	3.151
7	48	https://www.google.com/search?q=maldives+accomodation	Core	0.308
7	17	http://worldpopulationreview.com/countries/maldives-populat...	Core	0.308
7	34	https://www.google.com/search?q=Interesting+places+in+mal...	Core	0.585
7	40	https://www.google.com/search?q=MLA+Luqa+Airport	Core	0.74
7	11	http://www.mapsofworld.com/maldives/maldives-location-ma...	Core	0.861
7	12	http://www.where-are-the.com/maldives.php	Core	0.584
7	18	https://www.google.com/search?q=maldives+population+2014	Core	0.862

Fig. 1: PiMx Interface

data by different time windows (e.g. last hour, last 4 hours, today, yesterday etc.), as well as by application and file-type.

The *PiMx-Viz* is an interactive component inspired by the SessionGraph described in [10]. This provides visualisations of the unadulterated activity-graph as it evolves over time and the task-clusters as they are generated by the applied algorithm. The size of the nodes is relative to the number of accesses and it is also possible to click on each node separately and to visualise the induced subgraph generated by the nodes' neighbourhood.

The *PiMx-Stats* component was inspired by graphical tools such as Visone[5] and Gephi[6]. It presents a number of graph related statistics, such as the number of vertices and edges, the clustering coefficient, the average distance and diameter of the graph. There is also information about the number of search and removed nodes. Search nodes represent the pages associated with a search engine query. The relevant query and the number of times that this search node was accessed are displayed. Information about the type and number of occurrences of the events that were triggered is also provided.

Through the *PiMx-Clustering* view it is possible to view the details of the documents pertaining to each task-cluster. Each cluster is assigned a unique ID and each document in a cluster has associated with it a ranking value and information about the status generated by the algorithm. More details about this algorithm are found in Sect. 5

5 Incremental Graph Clustering Approach

In this section we give an overview of the incremental density-based task clustering approach that we've adopted. The scope behind our algorithm **iDeTaCt**

[5] `http://visone.info/`
[6] `http://gephi.github.io/`

is two fold: (i) identify those nodes that belong together in a task-cluster and (ii) identify when a switch between two nodes is effectively a task-switch.

Clustering entities into dense parts allows for the discovery of interesting groups in different networks. Furthermore, clustering on time-evolving networks is still an open research problem that has been addressed through different approaches including incremental clustering [2] which tracks the granular dynamics of a network, such as edge and node addition and deletion, rather than a time-window. This approach is quite applicable to the dynamics of information-seeking behaviours, whereby new documents are added over time, which in turn need to be assigned to an existing or new task-cluster.

5.1 iDeTaCt: incremental Density-based Task Clustering

The density-based clustering algorithm DBSCAN proposed by [6] produces partitional clustering, whereby a cluster is considered to be a continuous area of arbitrary shape that is denser than its surroundings. DBSCAN relies on the idea that the neighbourhood of a node up till some given radius ϵ defines the *"importance"* of that node. Nodes that have a minimum number, η, of other nodes at a distance less then ϵ are termed as *core nodes*. On the other hand, a node that has no such neighbourhood is given the status of *noise node*, unless it is contained within the neighbourhood of a core node, in which case it is assigned the status of a *border node*. Thus ϵ and η ensure that node neighbourhoods are dense areas.

In our case, we consider that a switch between two windows initiates an association between them, and this increases as more switches are effected. This incremental nature of the data can be represented by a graph, $G_a(V, E)$ that evolves with the introduction of new nodes (documents) and edges (switches). The edge weights represent the association between the nodes.

The clustering of those documents that pertain to the same task can also be represented by a graph, $G_c(V, E)$ that will also need to be updated incrementally, since a switch to a new document will trigger a decision process do deal with the change. The changes to $G_c(V, E)$ that our clustering algorithm has to deal with include:

i. the *creation* of a new cluster: when the association between two nodes exceeds the ϵ threshold;
ii. *merging* of two clusters: when either a core or border node in one cluster gets strongly associated with another core or border node in another cluster;
iii. *absorption* (a growing cluster): when the association between a core or a border node and a new node exceeds the threshold ϵ.

Consider a typical situation whereby the initial edge weight between two nodes u and v is $> \epsilon$. With increased switches, the association strength increases since the edge weight will decrease and possibly become $\leq \epsilon$. At this point, either, or both, of u and v can become core nodes (depending on η) with the consequence that nodes in their neighbourhood can either change status as well, form a cluster or merge with an existing one. It might also be the case that either u or v, or both, become border nodes, and thus form a potential cluster.

In **iDeTaCt** we use an association edge-weighting function $W : \mathbb{R} \to \mathbb{R}$ that maps the number of window-switches between two documents to an edge weight $w(u, v)$ in $G_a(V, E)$. We do not consider the direction of the edge, that is, an edge from node A to node B is considered the same as an edge from B to A. The resulting edge weight is inversely proportional to the number of window-switches. Thus a high number of switches will result in a lower edge weight. This is similar to the *proximity* and *influence* functions used in [7, 8] respectively, whereby two nodes are considered to be closer together if the edge weight between them is less.

We compute the number n of edges between two nodes as a fraction of a defined maximum number of edges, h. This maximum number is empirically set to 10 which is considered to be sufficiently indicative of a strong association between any two documents. Thus if the number of edges is 1, the value passed on to the W would be equal to $\frac{1}{10}$.

The edge-weighting function $W(\frac{n}{h})$ is based on the Epanechnikov kernel [8] and is defined as:

$$W(x) = \begin{cases} \frac{3}{4}(1 - x^2) & |x| \leq 1 \\ 0 & \text{else} \end{cases} \tag{1}$$

iDeTaCt takes as parameters the newly generated edge e and the old clustered graph $G_c(V, E)$ and works as follows:

– The association edge-weighting function *computeAssociation* takes as parameter the number of switches between u and v and returns the updated weight $w(u, v)$ of e.
– This weight is used to increase the ranking of nodes u and v within a cluster. If $w(u, v)$ is less than or equal to ϵ the node's ranking is increased by a factor of 0.85, otherwise it is increased by a factor of 0.15. This is in line with the way that Firefox's frecency algorithm[7] assigns a bonus to recently viewed pages.
– If edge e does not exist in G_c then e is added and $G_c(V, E)$ is updated. This results in endpoints u and v of e becoming connected in $G_c(V, E)$.
– Then for both u and v, if they are not core, we consider all their incident edges to check whether the changes have effected their status.
– If there are η or more such edges incident on node u then its status is set to *core*.
– If u is not *core* but is adjacent to a *core* node then its status is defined as *border*.
– Nodes that are neither *core* nor *border* are considered as *noise* and are placed on a stack for later consideration.

[7] `https://developer.mozilla.org/en-US/docs/Mozilla/Tech/Places/Frecency_algorithm`

- Then **iDeTaCt** calls **updateClusters** which performs a Breadth-First-Search over G_c to find the updated induced subgraphs. Each subgraph represents a cluster.
- In the process, **updateClusters** tries to include particular nodes from the stack that are still labelled as *noise* using the procedure *findWeakNodes*.
- In *findWeakNodes*, if a node i is found to be a neighbour to u and v in $G_c(V, E)$ which have a status of core than the status of i is changed to *weak* and it is added to $G_c(V, E)$. Although such nodes have a weak relation with the surrounding nodes, in that they fall short of the ϵ threshold, they are connected to nodes which in turn are strongly connected.
- **iDeTaCt** returns a list of clusters that can be visualised through the *PiMx-Viz* and the *PiMx-Clusters* components.

6 Evaluation

In our evaluation we wanted to verify whether **iDeTaCt** was able to: (i) identify those nodes that belong together in a task-cluster and (ii) identify when a switch between two nodes is effectively a task-switch.

For the evaluation we made use of the *PiMx* framework to simulate the task execution trails of the users from groups 1 (considered as the baseline group) and 2 (interleaved tasks). Details about the number of pages visited and the number of switches made by participants in these two groups can be seen in Fig. 2 and Fig. 3. We did not use the data from Group 3 since we wanted to initially evaluate our approach on data that was collected during a single session.

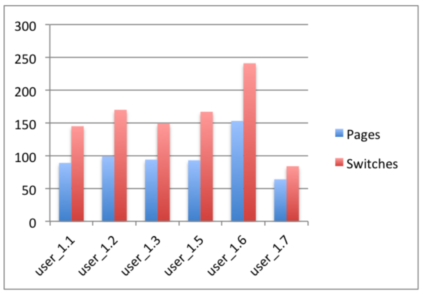


Fig. 2: Pages/Switches for Grp 1

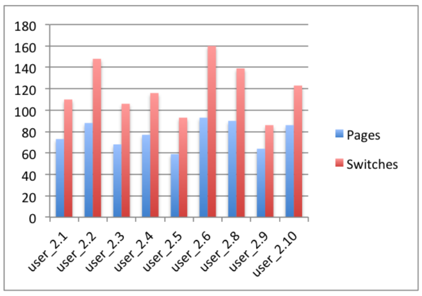


Fig. 3: Pages/Switches for Grp 2

For each trail we used **iDeTaCt** with core parameter η values of 1, (*D_1*) and 2, (*D_2*). With $\eta = 1$, two nodes will become core when they are connected by a single edge whose weight $w(u, v) \leq \epsilon$. Similarly with $\eta = 2$, a node will need to be connected to two other nodes through two similar edges. ϵ was set to the maximum value of 0.72 which is equivalent to a minimum of two window switches (using equation Eq. 1).

We used standard information retrieval metrics to evaluate the clusters for each of the three tasks. These metrics involve (i) *precision*, defined as the percentage of documents correctly assigned to a task-cluster over the total number of documents in the task-cluster, (ii) *recall*, defined as the percentage of documents correctly assigned to a task-cluster over the total number of documents that should have been assigned to that task-cluster, and (iii) *F1-measure*, defined as the combined measure that assesses a trade off between *precision* and *recall*. Whenever the algorithms generated two or more clusters for documents from the same task, we considered the cluster which was more representative of the task, that is, it contained the highest number of documents. In the case of the interleaved tasks in Group 2, we expect the algorithm to be able to cluster the interleaved tasks as if they were actually none interleaved.

Precision was 100% when we tried both *D_1* and *D_2* on all the task-trails from Group 1. However when we used *D_1* on the dataset from Group 2 it was less then 100% in all cases except one. When we changed η to 2 on the data from Group 2 we got 100% precision in 66% of the cases, in all the 3 tasks. In the rest, the precision was less than 100% for only one of the tasks.

We focus on the more interesting recall and F1-measure. The averaged results are shown in Fig. 4 and Fig. 5 respectively. We again compute the recall and F1-measure for all the task-trails from Groups 1 and 2, and we do this for every task separately.

Recall for the task-clusters generated on the data from Group 1 was highest when we used *D_1*, with the averaged recall being highest for task 2, at 70.7%. Task 3 had the lowest averaged recall at 36.5% due to the algorithm generating multiple, 2 or 3-node clusters. The possible reason for this could be due to the familiarity of the participants with this topic. The fact that a very important music event was forthcoming when the experiment was conducted, might have effected the participants' information seeking behaviour with many of them knowing where to search and thus the number of re-visits was low.

The F1-measure for the task-clusters from Group 1 was consistent with the recall and was again highest when we used *D_1*. As expected, the number of captured nodes with this value of η was higher than with *D_2* and the resultant task-clusters where denser, yet still separate. Task 3 had once again the least averaged F1-measure irrespective of η.

We now consider the recall and F1-measure based on the data from Group 2. The values obtained for the task-clusters associated with task 2 were the highest, and ranged between 50% and 60%. This trend is in line with the results we got for the task-clusters from Group 1, however both values for task 3 are slightly higher then those for task 1 except for the F1-measure based on *D_1*.

The recall and F1-measure based on the data of 2 of the participants from this group resulted in exceptionally low values for the task-cluster related to task 3. This was independent of η. For another user from the same group, we observed the same low values for the task-cluster related to task 1. The common feature observed across the data of these 3 participants was that the relevant task was fragmented in multiple 2 or 3-node clusters.

From the generated graphs it was possible to observe that all the participants tend to open up a number of documents which they only visit once. This was more accentuated in almost 50% of the cases from Group 2. For some of these, we manually inspected their log file and found that in fact these participants typically opened up a number of tabs in succession, as in the case of a result page associated with some query. They however only visited some of those opened tabs once. Different individuals however did revisit some of the documents multiple times and these acted like hubs/authorities to the other accessed documents thus allowing for the generated clusters to be more consistent.

The fact that **iDeTaCt** managed to cleanly separate all the tasks, using the data of all the participants from the baseline group is already encouraging. It is even more encouraging when we used the interleaved tasks and found the algorithm to be sufficiently reliable in more than 50% of the cases, even though this was at the expense of capturing less documents from the referenced tasks.

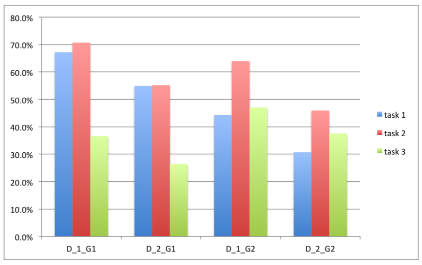


Fig. 4: Average Recall

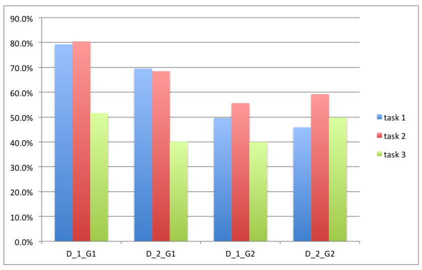


Fig. 5: Averaged F1-measure

7 Future Work and Conclusion

We intend to take into consideration the type of edge, which requires us to extend *iDeTaCt* to handle navigational events, especially those relating a query with its result pages. Navigational events accounted on average for 30% of all switches performed by each participant. We also plan to consider the window captions and apply a similarity function, such as cosine similarity, over this content, based on the clusters generated by *iDeTaCt*. Furthermore, we want to make use of *iDeTaCt* for task-resumption and in-line task/document recommendations. We intend to evaluate these extensions and compare our results.

In this paper we described the experiment we conducted to collect data for the evaluation of our *iDeTaCt* incremental graph clustering algorithm. We executed the algorithm over the task collections and used the *PiMx* framework to analyse its performance. The results showed a high precision and recall over the data from the baseline group and a recall of more than 50% over the data for the interleaved tasks. This is considered to be very encouraging and motivates us to further investigate how to improve our algorithm.

References

1. Bernstein, M., Shrager, J., Winograd, T.: Taskpose: Exploring Fluid Boundaries in an Associative Window Visualization. In: 21st ACM Symposium on User Interface Software and Technology, pp. 231-234. ACM Press New York, NY, USA (2008)
2. Charikar, M., Chekuri, C., Feder, T. and Motwani, R.: Incremental clustering and dynamic information retrieval. In Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (STOC '97), pp. 626-635. ACM, New York, NY, USA (1997)
3. Costache, S., Gaugaz, J., Ioannou, E., Niederee, C., Nejdl, W.: Detecting contexts on the desktop using bayesian networks. In: Desktop Search Workshop co-located with SIGIR (2010)
4. Dabbish L., Mark, G. and Gonzlez, V.M.: Why do I keep interrupting myself?: environment, habit and self-interruption. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11), pp.3127-3130. ACM, New York, NY, USA (2011)
5. Dumais, S., Cutrell, E., Cadiz, J.J., Jancke, G., Sarin, R., and Robbins, D.C.: Stuff I've seen: a system for personal information retrieval and re-use. In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval (SIGIR '03), pp.72-79. ACM, New York, NY, USA (2003)
6. Ester, M., Kriegel, H.-P., Sander, J., Wimmer, M., and Xu, X.: Incremental clustering for mining in a data warehouse environment. In Proc. of 24th VLDB Conference (1998).
7. Falkowski, T., Barth, A. and Spiliopoulou, M.: DENGRAPH: A Density-based Community Detection Algorithm. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI '07). IEEE Computer Society, Washington, DC, USA, 112-115 (2007)
8. Günnemann, S. and Seidl, T.: Subgraph Mining on Directed and Weighted Graphs. In Proc. of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2010), pp. 133-146. Hyderabad, India. Springer - Heidelberg, Germany. (2010)
9. Jones, W.P., Teevan, J.: Personal Information Management. ISBN 9780295987378, University of Washington Press (2007)
10. Mayer, M.: Visualizing web sessions: improving web browser history by a better understanding of web page revisitation and a new session- and task-based, visual web history approach. PhD thesis, University of Hamburg (2008)
11. Morris, D., Ringel Morris, M., Venolia, G.: Searchbar: a search-centric web history for task resumption and information re-finding. In: 26th annual SIGCHI conference on Human factors in computing systems, CHI '08, pp. 1207-1216. ACM Press, New York, NY, USA (2008)
12. Oliver, N., Smith, G., Surendran, A.C.: SWISH: Semantic Analysis of Window Titles and Switching History. In: 10th International Conference on Intelligent User Interfaces, pp. 194-201. ACM Press, New York, NY, USA (2006)
13. Wang, Q. and Chang, H.:Multitasking bar: prototype and evaluation of introducing the task concept into a browser. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10). ACM, New York, NY, USA, 103-112. (2010)