Fast and reliable inference of semantic clusters

Nicolas Fiorini, Sébastien Harispe, Sylvie Ranwez, Jacky Montmain, Vincent Ranwez

Fast and reliable inference of semantic clusters

Nicolas Fiorini[a, *], Sébastien Harispe[a], Sylvie Ranwez[a], Jacky Montmain[a], Vincent Ranwez[b]

[a] LGI2P research center from the École des mines d'Alès, Site de Nîmes, Parc scientifique G. Besse, 30035 Nîmes cedex 1, France
[b] UMR AGAP, Montpellier SupAgro/CIRAD/INRA, 2 place P. Viala., 34060 Montpellier cedex 1, France

A B S T R A C T

Document Indexing is but not limited to summarizing document contents with a small set of keywords or concepts of a knowledge base. Such a compact representation of document contents eases their use in numerous processes such as content-based information retrieval, corpus-mining and classification. An important effort has been devoted in recent years to (partly) automate semantic indexing, i.e. associating concepts to documents, leading to the availability of large corpora of semantically indexed documents. In this paper we introduce a method that hierarchically clusters documents based on their semantic indices while providing the proposed clusters with semantic labels. Our approach follows a neighbor joining strategy. Starting from a distance matrix reflecting the semantic similarity of documents, it iteratively se- lects the two closest clusters to merge them in a larger one. The similarity matrix is then updated. This is usually done by combining similarity of the two merged clusters, e.g. using the average similarity. We propose in this paper an alternative approach where the new cluster is first semantically annotated and the similarity matrix is then updated using the semantic similarity of this new annotation with those of the remaining clusters. The hierarchical clustering so obtained is a binary tree with branch lengths that convey semantic distances of clusters. It is then post-processed by using the branch lengths to keep only the most relevant clusters. Such a tool has numerous practical applications as it automates the organi- zation of documents in meaningful clusters (e.g. papers indexed by MeSH terms, bookmarks or pictures indexed by WordNet) which is a tedious everyday task for many people. We assess the quality of the proposed methods using a specific benchmark of annotated clusters of bookmarks that were built man- ually. Each dataset of this benchmark has been clustered independently by several users. Remarkably, the clusters automatically built by our method are congruent with the clusters proposed by experts. All resources of this work, including source code, jar file, benchmark files and results are available at this address: http://sc.nicolasfiorini.info.

Keywords:
Clustering
Cluster labeling
Semantic indexing
Neighbor joining
Complexity analysis

1. Introduction

In recent years, there has been a great evolution in document analysis applications. Clustering is the lion's share of automated document processing since it provides relevant organization of documents that synthesizes and underlines their properties and meanings. In our everyday life, organizing documents in (sub)folders is such a recurrent and crucial task that we tend to forget how tedious and time consuming it is. However, at some point we all complain about the emails piling up, the unsorted holiday pictures, the web pages that we saved on our bookmark list but that we cannot find among the hundred other ones. As more and more documents are now semantically tagged, it should be possible to automate their semantic organization, providing tools that would, for instance, organize your local library of scientific papers based on their keywords and MeSH annotations, structure your bookmarks based on the metadata of the corresponding web pages or sort your emails based on the tags you associate to them while accounting for the fact that "conference" and "workshop" tags are both refinement of a "scientific meeting" tag.

Conceptual annotation, also called semantic indexing, is proposed as a generic way of representing the content of documents. It aims at summarizing document contents with a small set of keywords or concepts of a knowledge representation such as an ontology. Such a compact representation of document contents eases their use in numerous automated processing tasks: content-based information retrieval, corpus-mining or documents classification. Semantic indexing is as tedious as complex: a synthetic and relevant semantic annotation requires a good understanding of the

* Corresponding author at: NCBI: National Center for Biotechnology Information, National Library of Medicine, Bldg. 38A rm 8N811B, 8600 Rockville Pike, Bethesda, Maryland 20894, USA.
E-mail address: nicolas.fiorini@nih.gov (N. Fiorini).

subject area the documents refer to, as well as a deep familiarity with the chosen knowledge representation. In recent years, an important effort has been devoted to automate semantic indexing, leading to the availability of large corpora of semantically indexed documents.

In this context, this paper introduces a hierarchical agglomerative clustering method that is based on the conceptual annotations of documents. The first originality of this approach lies in the use of a groupwise semantic similarity measure as the metric for document similarity: two documents are said to be close when the concept sets annotating them are similar in the sense of the semantic measure. Its second originality is that the similarity of two clusters is estimated as those of two documents rather than by some sort of initial document similarities aggregation as usually done. In our approach, when agglomerating two clusters, the conceptual index of the resulting larger cluster is automatically computed and then used to determine its similarity to others clusters consistently with the proposed cluster hierarchy. The final originality of our method lies in this tide imbrication of the clustering and labeling tasks which guarantees their consistence. The semantic similarity among clusters can hence be represented as tree branch lengths in the tree representation of the clustering thus underlining semantic properties of those clusters that can be used to identify the most meaningful clusters. To this aim, we propose a post-processing of the cluster hierarchy that takes advantage of branch lengths heterogeneity in the tree to only keep the most meaningful clusters.

The paper is organized as follows. The next section gives the context and our positioning with respect to the literature. Section 3 details the method and provides its space and time complexities. Section 4 presents the evaluation protocol. Section 5 provides the results obtained on a benchmark and their comparison with end-users' clusters; it discusses them and opens some perspectives. Finally, conclusions are drawn in Section 6.

2. Related work

Clustering is central to many applications in very different fields where people want to analyze and compare documents in the light of the domain knowledge they belong to: genes indexed by the Gene Ontology, scientific papers indexed by MeSH terms, bookmarks or pictures indexed by WordNet to cite a few. Organizing these documents manually within a hierarchy of named clusters/folders is a tedious everyday task.

As most operating systems use a hierarchical folder structure to organize electronic documents, people are now very familiar with this kind of document/folder organization. This organization provides a hierarchical representation of documents as they are grouped within imbricated named folders that can be seen as labeled clusters (each folder being a cluster labeled by its name). We all have faced the limit of such an approach. Especially when dealing with new documents that do not properly fit in the current hierarchy and would thus require to completely reorganize it, or when dealing with documents that could indifferently be placed in different folders. It has thus been suggested to replace the hierarchical folder approach by a document tagging system. An extensive comparison of those two approaches is provided in [1]. It concludes that there is no clear winner and that both strategies have pros and cons. For instance the folder strategy allows to declutter mailbox whereas multiple tag approaches facilitate later document search and allows to reveal unexpected or forgotten document connections. Tag approaches have been popularized by websites such as Twitter and its well known hashtag system. Numerous softwares have recently evolved to let users easily add multiple tags to documents and it is now possible to tag most document types using everyday life software applications. However recent work seems to indicate that, for most tasks, end-users continue to favor folder-based organization over tag-based one [2]. One can argue that this may be due to the force of the habit but this does not change the fact that end-users tend to favor hierarchical organization of their documents despites the advantages of the tag approach in certain cases. Here we propose to give the user the benefit of both approaches, while removing the tedious task of (re)constructing the first draft of their folder hierarchy by automatically building this hierarchy based on the document tags.

The whole point of clustering is to find groups of similar items that are different from other groups. Clustering methods are numerous [3] and depend on the type of data processed as well as clustering requirements and objectives that are fixed. In this paper we only consider hierarchical clustering methods, which produce a hierarchical representation of items instead of a flat partition of those items. The standard bottom-up strategy for this task processes by considering initial documents as singleton clusters and repeatedly grouping the two closest clusters into a new one, hence reducing by one the number of current clusters, until only one cluster remains.

The two main features of hierarchical agglomerative clustering are (i) the distance measure used to compare singletons and (ii) the approach used to distinguish which clusters are the closest ones. The distance measures implicitly determine the features used to cluster the documents. However, note that several distance measures can rely on the same features, but using them differently and hence providing a great diversity of output clusters. In this work, we focus on clustering approaches that rely on document semantic metadata. More precisely, we consider that each document is annotated by a set of concepts that are organized in hierarchical structures (e.g. ontologies, thesaurus) and that the clustering relies solely on these semantic annotations. Many semantic measures have been proposed to estimate the semantic similarity of two concepts [4], some relying only on the underlying hierarchical structuture (e.g. the path length between the compare concepts, *Information Content (IC)* of the compared concepts, etc.) whereas others use additional information such as color spectrum for image or word count for text documents. Note that the IC of a concept could also be defined solely based on the position of the concept within the ontology (roughtly speaking the closer the concept is to the ontology root the more generic it is and hence the lower is its IC) or it could be defined using additional external information such as the frequency of the concept in a representative corpus.

2.1. Semantic clustering

The literature presents some methods called semantic or mixing clustering with ontologies or metadata, however, there is no proper consensus on a field called semantic clustering the same way we define it, which is clustering documents by using their semantic descriptions. Kuhn et al. [5] for example introduced the concept of semantic clustering as the fact of grouping documents containing the same vocabulary. This approach is called semantic as they try to capture the meaning of the documents to cluster them, which is quite different from our objectives of clustering documents based on their semantic indices. Clerkin et al. [6] propose to use clustering in order to discover and create ontologies — and not using ontologies to cluster documents.

Some other studies are nevertheless closer to the scope of our work. Some researchers have for instance studied the impact of integrating knowledge base information in clustering algorithms [7]. To the best of our knowledge, Hotho et al. [8–10] have been the first to consider this kind of approach. Their work consists in enriching document annotations with background knowledge — in the most recent part, WordNet. Everything starts with the association of each document with a vector of term frequencies, further referred to as term vector. After being altered based on

WordNet information this vector is used as input of a k-means clustering approach to organize documents using term vector similarities. During the alteration the initial values of the term vector may be filtered to ignore low frequency terms. The Wordnet concepts related to the vector frequent terms (and some of their hypernyms) may also be added to enrich this vector. Their overall conclusion is that relying on conceptual descriptions of documents improves the quality of the proposed clusters. The authors explain that this improvement is due to the relationships between concepts (and thus the presence of common hypernyms, in their approach), where classical text clustering lacks such relationships. For example, they show that documents about COFFEE and CACAO were gathered in a FOOD cluster while food was never literally mentioned in those documents. Baghel and Dhir [11] propose a variant of this approach also based on WordNet. They also rely on concept frequency vectors that are mapped from texts, but they use them as input of a hierarchical clustering approach instead of a k-means one. Breaux and Reed [12] as well as Sedding and Kazakov [13] propose slightly analogous techniques relying on variants of hierarchical and k-means clustering algorithms, respectively.

Spanakis et al. [14] propose to use Wikipedia to test their novel Conceptual Hierarchical Clustering (CHC). They suggest a richer model for representing documents not only using the frequency of occurrence of each concept but also consider other features such as the position of the concept within the document or the number of links to the corresponding Wikipedia webpage. CHC is a hierarchical clustering technique that then relies on a combination of their various Wikipedia based features.

Song et al. [15] are the first to use semantic similarities as a metric for clustering semantically annotated documents. This work features one of the ideas proposed in [10] of mapping the term vectors to all the concepts they may refer to. The clustering method they proposed is quite uncommon compared to existing ones as it relies on a genetic algorithm, a classical meta-heuristic that searches the solution space by altering and combining a set of candidate solutions using principles analogous to natural selection and heredity.

The use of knowledge for enhancing clustering has also been successfully tried for clustering non-text entities that have been semantically indexed, e.g. genes [16,17]. Transcriptomic expression analysis (e.g. using DNA array) deals with a large number of expression measurements per gene. The clustering of genes, according to their expression profiles, can benefit from gene ontology annotation. Indeed, Liu et al. [16] show that using the annotations to prune the search space — so that irrelevant solutions are not explored — greatly reduces the computation time of the algorithm while producing results somewhat comparable in terms of cluster quality.

The contributions described in this section are more or less related to what we call semantic clustering, ontology-based clustering or ontology-driven clustering. Although it is certain that relying on an ontology helps clustering documents, few works detail how to rely solely on the document annotations and the underlying ontology. For example, Nasir et al. [18] propose an approach that extracts semantic relatedness in order to cluster textual documents. Figueroa and Neumann [19] aim at semantically classifying queries based on the — textual — contexts they extract from search sessions. Liu et al. [20] evaluate their results using two datasets, of which one is unavailable and the other one is the famous 20 Newsgroups[1], a textual dataset. Finally, the approach detailed by Zhu et al. [21] seems to be much closer to our work because it uses the MeSH concepts annotating biomedical documents, but they do not provide the list of document identifiers that they use to evaluate the method. It is thus impossible to fairly compare our work to any of these approaches, since there is no case in which the source code or intermediate files (e.g. those containing the conceptual annotations) are provided. Comparing to these approaches would thus require to add a step for extracting concepts from the texts — which is a whole domain intrinsically —, that can create a bias for evaluating the sole semantic clustering.

Besides, hierarchical clustering has been barely explored when considering ontologies e.g. [12,22] compared to other methods like k-means. Semantic similarities have been exploited in few works [15,22,23], but they did not consider the existing and numerous semantic similarities except in [24], which focuses on genes. Unfortunately, this work has two downsides. First, it is only applicable to genes, although the core idea is actually generic. It relies on GO (Gene Ontology) concepts — so it cannot load another ontology — to cluster the genes based on the semantic similarity of their annotations. Then, they map the clustering with the expression data as it is meant to be used for gene clustering after a microarray analysis. Second, they rely on a basic hierarchical clustering algorithm with classical linkage functions instead of exploring the behavior of computing semantic similarities during the clustering (see Section 3.2 for more details on this aspect).

2.2. Semantic cluster labeling

In analysis tasks and user interfaces, clusters have to be rapidly understood by the users. In Information Retrieval, for a query "travel to Germany", the results can be presented as clusters of hotels, restaurants, sightseeings, etc. Another common use of cluster labels is when we want to understand how items have been gathered [25]. After clustering genes by using their expression data in a few environments for example, one may be interested in what characterizes each group. Consequently, cluster labeling often follows a clustering step as it is an important task for cluster analysis [26].

Role and Nadif [27] state that in most labeling approaches after text clustering, labels are simply terms picked from the texts according to their frequency. The limitation of such a process is that relationships among the words are not represented. They thus propose an approach to make a graph representation of terms for the clusters that gives the user a better understanding of the cluster meanings.

As for clustering, the benefit of using external resources such as Wikipedia has been predicted and tested [28]. Authors realized that even if the gold standard words were present in the documents of a given cluster, they would rarely be selected to annotate this cluster. In order to have better labels, they thus propose to define for each cluster a list of candidate terms (by relying on their previous work [29]) then query Wikipedia using those terms and use the metadata of the returned Wikipedia articles to annotate the cluster. The results show a significant improvement compared to simple term extraction based on frequencies for instance.

Most of the work regarding semantic cluster labeling is related to gene clusters. The reason is that the scientific community working with the Gene Ontology (GO) is one of the most dynamic ones regarding the use of knowledge representation for data analysis. Genes being annotated with GO, the concepts of this ontology can be used for automatically labeling inferred gene clusters. GOstat [30] proposes to annotate a group of genes by finding overrepresented GO concepts among their annotations. This approach takes the ontology structure into account as ancestors of concepts annotating the genes are considered to be potential labels. Several other tools add slight variations such as different statistical models, multiple test corrections [31,32], a better scalability [33], a broader range of species [34] or new visualization tools [35–37].

[1] http://qwone.com/~jason/20Newsgroups/

Although these approaches are relevant in a biological context, the suitability of the way GO concepts are picked for labeling the clusters in another context can be discussed. Overrepresentation is one way of synthesizing a set of concepts by picking those that directly or indirectly appear the most. This kind of approach, however, does not consider the specificity or genericity of the labels. Especially when labeling hierarchical clusters, the specificity/genericity ratio needs to be controlled in order not to give the same labels to successive parent nodes. Besides, the most elaborated approaches for cluster labeling are related to GO while this task is useful in many contexts with various ontologies. In essence, these methods could certainly be adapted to work with another ontology, however, the papers are often in application notes format that do not provide implementation details and the source code is rarely available.

Finally, several studies have been proposed to ease document annotation. Among all indexing models, Yang [38] and Trieschnigg et al. [39] both stated that the k-Nearest Neighbors (k-NN) approach is the only method that can scale while providing good results. This is a two-step approach, first it identifies the neighbors of the document to be annotated (e.g. based on co-authoring or Natural Language Processing) and second it summarizes those neighbor annotations to build a candidate annotation of the document to be annotated. Note that this second step is similar to building an annotation of the document cluster, i.e. the cluster is made of the considered neigbors. Methods designed for this second step can thus be used for cluster annotation. Among existing solutions, the approach developped for the User-oriented Semantic Indexing (USI) tool [40] fits well in this context as it clearly focuses on the annotation summary step. It also implements an explicit objective function so that it can easily be adapted to better fit hierachical clustering annotation specificity. The objective function of USI is designed to search for an annotation that is as concise as possible while being as close as possible to the considered neighbor document annotations.

Document clustering and semantic cluster annotation are so far considered as two successive, and rather independent tasks mostly because the clustering can be done based on different information types. However when the clustering is based on the document semantic annotations the two tasks could probably benefit from one another and should probably be done simultaneously to ensure the overall consistence of the final output. As far as we know our approach is the first to propose a really tight imbrication of the clustering and labeling process.

3. Methods

This section details the solution we propose to hierarchically cluster documents using their semantic indices while providing semantic labels of the resulting clusters. Note that for complexity analysis we assume here that the ontology has already been parsed and that pairwise similarities of concepts as well as concepts' ancestors/descendants have been pre-computed once and for all. These can then be retrieved efficiently in $\mathcal{O}(1)$. As this assumption is done for both the baseline clustering algorithm and our semantically aware clustering variant this does not bias the complexity comparison of those two methods.

3.1. Hierarchical clustering principles

The hierarchical clustering, or equivalently the tree representing it, can be obtained by following two strategies: an agglomerative or a divisive strategy. The construction is said bottom-up for an agglomerative strategy — the tree is built from the leaves to the root — and top-down for a divisive strategy. Top-down approaches require for each iteration to find the most distant pair of clusters among a list of 2^n clusters. Usually, top-down approaches use a flat clustering technique such as the k-means as a subroutine to keep a polynomial time complexity. Note that in our case, using k-means is not straightforward as it would require numerous estimations of the centroid of a cluster of indexed documents. Such centroids could be provided using the median annotation of the document of the considered cluster, using USI, but this would be computationally costly especially for large clusters.

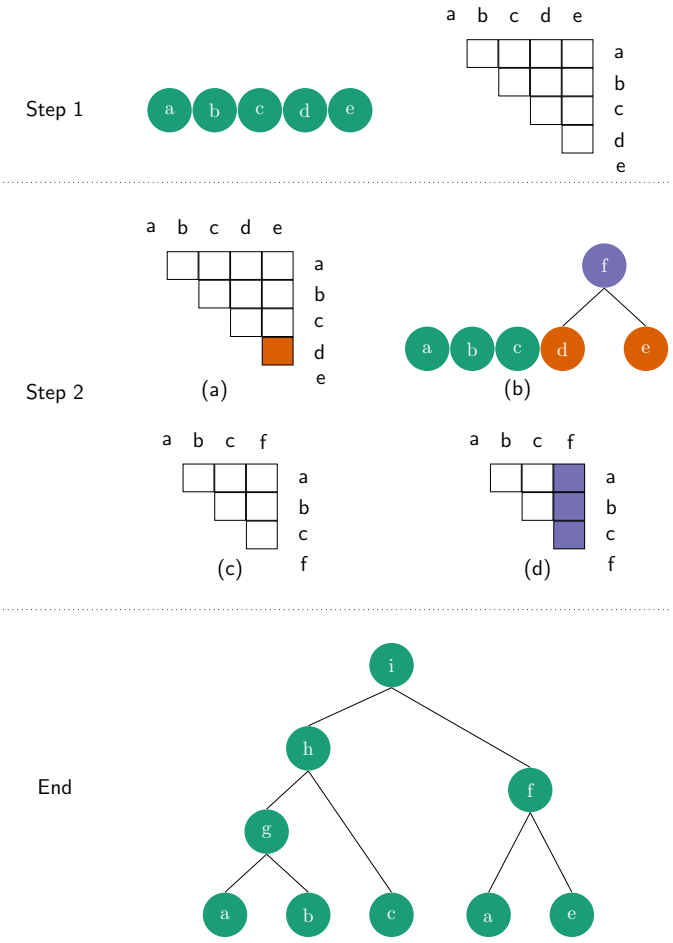

Fig. 1. The hierarchical agglomerative clustering. Step 1 initializes the clustering by creating cluster singletons and a pairwise similarity matrix. Step 2 consists in finding the closest clusters in the matrix (a), creating a new cluster f gathering them (b), updating the similarity matrix accordingly (c and d). Step 2 is repeated until only one cluster remains. At the end of the process, the obtained tree is given as a result, its topology summarizes the infered clusters and the branch lengths reflect the semantic distance among those clusters.

The bottom-up construction is conceptually easier as it only needs to be able to find the two closest clusters at each iteration. In total, we need to compare $\frac{n(n-1)}{2}$ pairs of clusters. Details of the method are presented in Fig. 1. There are two main steps in Hierarchical Agglomerative Clustering (HAC). One, the algorithm is initialized. Each document — more commonly called observations in the clustering community — to cluster is put in a singleton cluster and a pairwise similarity matrix of all singletons is computed. Two, the closest clusters in the matrix are identified (Fig. 1a) and gathered within a new cluster (b). The matrix is updated accordingly by removing the rows and columns associated to the two merged clusters and adding one row and column to store the similarities of the newly created cluster with others (c,d). This agglomerative step is repeated until there is only one remaining cluster, that is all clusters have been agglomerated. Note that branch lengths of the output tree can be used to reflect cluster similarity: the closer the

clusters, the shorter the branches connecting them. The tree structure then allows the users to grasp several clustering granularities at once.

The key feature of HAC to define is clearly the way to compare (sets of) observations to initialize and update the similarity matrix. In fact, HAC algorithms require the definition of two functions called the similarity metric and the linkage criterion. The former is used to fill the initial matrix of similarities (step 1). Depending on the kind of data on which the clustering is made, several existing functions can be used for this purpose. For example, Euclidian distance is appropriate for comparing numeric vectors and the Levenshtein distance for comparing strings.

When two clusters are agglomerated, the similarities of the newly created cluster with others are calculated by using the linkage criterion that defines how to compare two sets of observations. The choice of this function may impact the clusters shape and the branch lengths of the resulting tree. Most of the linkage criteria are function of the pairwise similarity metric that is used to compare the singletons. The average linkage *ALINK* is one of the most widespread:

$$ALINK(Cl_x, Cl_y) = \frac{1}{|Cl_x||Cl_y|} \sum_{x \in Cl_x} \sum_{y \in Cl_y} s(x, y), \quad (1)$$

where Cl_x, Cl_y are the two distinct clusters that are compared and $s(x, y)$ is the similarity metric used at step 1 of the HAC. Note that given $ALINK(Cl_x, Cl_z)$ and $ALINK(Cl_y, Cl_z)$, the value of $ALINK(Cl_x \cup Cl_y, Cl_z)$ can easily be obtained in $\mathcal{O}(1)$ — recall that clusters are disjoints. This highlights that the HAC complexity depends on the chosen pairwise measure complexity and linkage function. Say D is the set of documents to be clustered, each of the $\mathcal{O}(|D|^2)$ initial pairwise similarities can be computed in $\mathcal{O}(S^2_{max})$ for the considered semantic similarity measures as in [40], where S_{max} is the maximum number of concepts annotating a document. The HAC has $\mathcal{O}(|D|)$ agglomerative steps, at each step the current similarity matrix is browsed to find the best pair among $\mathcal{O}(z^2)$, where z is the current number of clusters, assuming that the linkage function used is a simple one (e.g. *ALINK*) that can be computed in $O(1)$, the agglomerative steps thus have a total complexity of $\mathcal{O}(\sum_{z=1}^{|D|}(z^2)) = |D|^3$. The overall complexity of this baseline HAC approach is $\mathcal{O}(|D|^2 S^2_{max} + |D|^3)$.

3.2. Semantic clustering

In the following section we thoroughly detail our approach considering the following key steps: (i) computation of the initial pairwise similarity matrix; (ii) annotation of a newly created clusters and update of the similarity matrix; and (iii) post-processing of the resulting HAC tree.

3.2.1. Computation of the initial pairwise similarity matrix

As explained above, the similarity metric is chosen according to the type of data to cluster, e.g. the Levenshtein distance is often used for clustering strings. It thus seems intuitive to rely on a semantic similarity measure as the elementary metric to build the similarity matrix of documents annotated by concepts. As in our previous work related to semantic indexing [40] we choose to rely on the Lin measure [41] for pairwise similarities and on Best Match Average (BMA) [42] for groupwise semantic similarities. Indeed, although many semantic similarities have been proposed to estimate the semantic similarity of a pair of concepts, they often are variations of the same general ideas [43]. Among the various existing variants we choose to use the Lin's measure which is one of the most widespread ones and have proved to be efficient to tackle related (semi-)automatic annotation tasks [44,45]. This measure is strictly based on the information contained in the graph provided by the knowledge base. While Resnik's semantic similarity measure [46] corresponds to the IC (see Section 2) of the common ancestor of the concepts that are compared, Lin's one refines it by also considering the IC of both concepts individually. Moreover, as documents/clusters are annotated by a group of concepts and not just a single one, we use the Best Match Average (BMA), a simple groupwise semantic similarity, to aggregate the pairwise similarities of annotations into a single value reflecting the overall semantic similarity of the compared elements. This metric estimates groupwise similarities in a rather intuitive way, because it consists of finding, for each concept of one group, the closest one in the other group and vice versa. It thus allows key algorithmic optimizations to search for the best cluster annotation in reasonable time [40]. All the semantic similarities corresponding to these identified pairs are averaged. More precisely, we have:

$$\begin{aligned} sim(Cl_x, Cl_y) &= sim_{BMA}(L_x, L_y) \\ &= \frac{1}{2|L_y|} \sum_{c \in L_y} sim_m(c, L_x) + \frac{1}{2|L_x|} \sum_{c \in L_x} sim_m(c, L_y), \end{aligned} \quad (2)$$

where L_x and L_y are the set of labels indexing the clusters Cl_x and Cl_y respectively; $sim_m(c, Cl_y) = \max_{c' \in Cl_y}(sim(c, c'))$ and $sim(c, c')$ is the Lin pairwise semantic similarity. In other words, sim_{BMA} is the average of all maximum similarities of concepts in Cl_x w.r.t. Cl_y and vice versa.

3.2.2. Cluster labeling

Once the highest value in the similarity matrix has been identified, the two closest clusters Cl_x, Cl_y are merged into a new cluster Cl_{new} and we need to define the labels L_{new} that characterize Cl_{new}. This can be done by using the labels L_i of documents D_i within this cluster L_{new}. A simple solution, retained in our baseline approach, is to merge the labels of those documents. Such a naive solution has however the drawback of leading to indices of growing size as the clustering goes on. In order to keep reasonable label sizes even for very large clusters, an alternative solution is to summarize the D_i annotations using a label set L_{new} which, while being similar to D_i annotations, consists of fewer concepts. Building such a summary annotation is exactly what USI have been developed for.

As the time complexity of USI strongly depends on the number of annotations to be summarized, the straightforward solution of considering the annotations of each and every document of Cl_{new}, that we denote heavy semantic clustering (or HSC in brief), is highly time consuming. We thus consider an alternative approach where the annotation of the new cluster is the USI summary of the labels L_x and L_y of the merged clusters Cl_x of Cl_y. Indeed, while USI was developed to annotate a document based on neighbor document annotations, in this case we want to annotate clusters based on the annotations of their sub-clusters. This strongly reduces the time complexity of this step as the number of annotations ("neighbors", for USI) is now always 2, regardless of the cluster size. We denote this variant LSC for Light Semantic Clustering.

Let us now move on from the objective function originally optimized by USI:

$$f_{USI}(L) = \frac{1}{|\mathfrak{L}_K|} \sum_{L_d \in \mathfrak{L}_K} \Big(sim_{BMA}(L, L_d) \Big) - \mu |L|. \quad (3)$$

$\mathfrak{L}_K$ is the set of neighbor document annotations, L_d is the set of labels of document d and μ represents a trade-off parameter between the expressivity of the summary and its concision. Note that the sub-clusters may have very heterogeneous sizes, and they are by construction more specific than the newly created one that encompasses them. To annotate Cl_{new} the cluster agglomerating Cl_x and Cl_y, we therefore rely on the following adaptation of the USI

objective function that better fits our needs:

$$f(L) = \frac{1}{W} \sum_{i \in \{x,y\}} \left(\ln(|Cl_i|) \times sim_{BMA}(L, L_i) \right) - \mu |L| \sum_{c \in L} IC(c), \quad (4)$$

$$with\ W = \sum_{i \in \{x,y\}} \ln(|Cl_i|), \quad (5)$$

Cl_i is one subcluster of Cl_new, L_i is its annotation and $IC(c)$ is the information content of concept c, as mentioned in Section 2. We now take the cluster size into account using a log-based weighting so that when merging two clusters, the labels of the largest one has more influence but do not obliterate the other cluster annotation. Imagine merging two clusters where the first Cl_x contains 20 documents that are all indexed by the same label set L_x, whereas the second cluster Cl_y contains a single document indexed by L_y; then we aim at annotating Cl_{new} by providing more weight to labels of L_x (as they index 20 out of the 21 documents of Cl_{new}) without masking the fact that Cl_{new} has two subclusters that are semantically heterogeneous and only one of those two subclusters contains documents indexed by L_x. In order to solve the problem stated in Eq. 4, we rely on exactly the same algorithm as USI (see [40]), where we only replace the calculation of the original objective function by the one detailed in Eq. 4.

3.2.3. Cluster postprocessing

The HAC algorithm per se produces a binary tree which is difficultly exploitable by a user. Grouping the clusters by pairs is legitimate for the algorithm complexity; however, a user would expect larger categories to appear instead of a dense binary tree. Consider for instance that three documents have the exact same index, then the HAC produces a first cluster C_1 with two of them and a higher level cluster grouping C_1 with the third document, whereas the end-user would expect a single cluster containing the three documents. Alternatively, consider a corpus made of 3 completely unrelated documents, once again the HAC produces a binary tree containing a cluster of two documents picked randomly whereas the end-user would expect the three documents to appear at the same level in the hierarchical clustering. The purpose of our post-processing algorithm is to detect such pattern in the tree output by our HAC. It relies on branch lengths that reflects the semantic similarity between the clusters: a cluster that is too close (case 1) or too far (case 2) from its children *and* parent is thus removed.

The threshold values used during this post-processing are automatically estimated based on the distribution of the HAC branch lengths. Indeed this distribution is often bimodal. As the distribution of distances may vary depending on the dataset, we define two quantiles α, β based on a training set. For example, with the training set of the benchmark proposed further (see Section 4), best results are obtained with $\alpha = 0.2$ and $\beta = 0.87$. Second, we define two threshold values th_l, th_h for the low and high values below or above which distances are considered for the post-processing tree flattening. A recursive function browses the tree with a top-down strategy. For each node it browses, if its parent *and* its children are both highly distant or both highly close, then it branches the children nodes to the parent one and removes the current node. Finally, the algorithm stops when it reaches the leaves and returns the processed tree by providing its root.

3.3. Time complexities of the proposed algorithms

We do not aim at ameliorating the generic HAC complexity in any way, but rather to explore the relevance of using HAC in tight conjunction with semantic similarity measures to provide more meaningful clusters. Nevertheless, as we want to provide a scalable method, we need to ensure our approach has a complexity comparable to standard HAC algorithms. To this aim, this section details the proposed algorithms we rely on and their complexities.

3.3.1. Algorithm details

The algorithm is a greedy algorithm leading to a binary tree inspired by the very common Neighbor-Joining method in systematic biology [47]. Algorithm 1 details the whole process. During the initialization, the first clusters and the tree are initialized: each cluster is a leaf and its labels are the annotation of the document it represents (l.5–9). The similarity matrix of trivial clusters is computed by using semantic similarities of document labels (l.10–15). Then, the agglomerative process begins and iterates until there is only one remaining cluster. Each iteration consists of finding the pair of closest clusters Cl_x, Cl_y in the matrix (l.18). A new cluster Cl_{new} is created (l.19). In order to label the newly created cluster we rely on USI that provides an efficient algorithm to synthesize the annotations of several documents [40] — in our case, the two children (l.20) of the new cluster. The new cluster is hence labeled

Algorithm 1: Clustering and labeling of a set D of documents.

```
1  Function Cluster (D, μ, S_max, θ)
       Input  : The set of documents to cluster D, a real number
                μ ∈ [0; 1], the maximum size of cluster labels
                S_max, an ontology θ
       Output: Hierarchy of clustered documents and cluster
                labels
2      if ∃d ∈ D, |L_d| > S_max then
3          print an error and exit;
4      end
5      clusters ← {};
6      for i ← 1 to |D| do
7          Create a childless cluster Cl_i with labels L_i = A_i;
8          clusters ∪ {Cl_i};
9      end
10     Create a matrix M of size 2|D| × 2|D|;
11     for i ← 1 to |D| do
12         for j ← i + 1 to |D| do
13             M(i, j) ← sim_g(L_i, L_j);
14         end
15     end
16     new ← |D| + 1;
17     while |clusters| > 1 do
18         Find the pair of remaining clusters Cl_x, Cl_y with the
           highest similarity;
19         Create a new cluster Cl_new;
20         K ← {Cl_x, Cl_y};
21         Create L_new ← USI_Annotate(K, μ, S_max, θ) the labeling
           of Cl_new;
22         Add child Cl_x to Cl_new with a distance of
           1 − sim_g(Cl_new, Cl_x);
23         Add child Cl_y to Cl_new with a distance of
           1 − sim_g(Cl_new, Cl_y);
24         clusters ← clusters ∪ {Cl_new};
25         clusters ← clusters \ {Cl_x, Cl_y};
26         for Cl_i in clusters do
27             i ← index of Cl_i in M;
28             M(i, new) ← sim_g(L_i, L_new);
29         end
30         new ← new + 1;
31     end
32     root ← clusters[1];
33     return root
34 end
```

by a synthesis of the annotation of its two children clusters using the variant of the USI objective function provided in Eq. 4. We also estimate and store the distance of the new cluster with each of its child by computing the semantic similarity of their labels (l.22–23). This step allows us to build a tree with branch lengths representing the semantic similarities of the nodes/clusters. The transformation of semantic similarities into semantic distances is done using the following equation, which assume that the semantic similarity measure is bounded in an interval [0; 1], as most of them are so [4]:

$$\begin{cases} d(Cl_n, Cl_x) = 1 - sim_g(Cl_n, Cl_x), \\ d(Cl_n, Cl_y) = 1 - sim_g(Cl_n, Cl_y). \end{cases} \tag{6}$$

Finally, the similarity matrix must be updated. The new cluster is added to it and all the similarities with other clusters are computed. The rows and columns corresponding to the two agglomerated clusters are removed (l.24–29).

3.3.2. Complexity analysis

Algorithm 1 features $S_{max} \in \mathbb{N}$ as an input that aims at bounding the maximal size of labels associated to the clusters. In practice, S_{max} has been set to 20 in the experiments detailed in Section 5.2. This is subject to variations depending on the dataset because if documents are heavily annotated, S_{max} should be higher, although a document is more likely to be annotated by few concepts in general. S_{max} is used as a parameter of the *USI_Annotate*($\cdot$) function. Note that the first lines of the algorithm guarantee that there is no inconsistency between D and S_{max}. Let us detail the few steps preceding the actual clustering of the documents. Creation of initial clusters (l.5–9) is in $\mathcal{O}(|D|)$. Filling the matrix requires to fill $\mathcal{O}(|D|^2)$ cells, each of which is computed in $\mathcal{O}(S^2_{max})$ for the BMA composite average. Initialization of the algorithm (l.2–15) is thus made in

$$\mathcal{O}(|D|^2 S^2_{max}). \tag{7}$$

The clustering process consists of $|D| - 1$ iterations during which three main processes occur. At each iteration, the matrix is browsed once to find the best cluster pair (l.18) in $\mathcal{O}(z^2)$ where z is the current number of clusters. The label of the cluster is then computed by using a modified version of the USI annotating algorithm for which the complexity is $\mathcal{O}(knS_{max} + n^3)$; where k is the number of document annotations to be summarized, n is the size of the set L_0 of candidate concepts and S_{max} is the maximum size of any annotation. For cluster labeling, $k = 2$ (the annotations of the two merged clusters, Cl_x and Cl_y), the initial set for labeling the cluster, denoted L_0, is made of the concept annotating Cl_x and Cl_y plus their first hypernyms. Indeed, to control the scalability of the algorithm, we introduce a hypernymy parameter h that limits the search among ancestors. That is, $anc(c, h)$ are the h direct ancestors of concept c. As a result, the search space size $n = |L_0|$ is in $\mathcal{O}(hS_{max})$. As $k = 2$ and n is in $\mathcal{O}(hS_{max})$, the complexity of the adapted USI algorithm is $\mathcal{O}(2(hS_{max})S_{max} + (hS_{max})^3) = \mathcal{O}(h^3 S^3_{max})$. The third and last important task is the computation of new semantic similarities with the $(z - 1)$ other clusters, which is done in $\mathcal{O}(zS^2_{max})$. Consequently, the complexity of each iteration in the while loop is a browsing of the matrix, a labeling of the new cluster and the computation of new similarities, leading to an overall complexity of each while iteration (l.18–29) of

$$\mathcal{O}(z^2 + h^3 S^3_{max} + zS^2_{max}). \tag{8}$$

The complexity of all iterations of the while loop, which dominates the one of the initialization and is thus the time complexity of the whole algorithm, is hence:

$$\mathcal{O}\left(\sum_{z=1}^{|D-1|} \left(z^2 + h^3 S^3_{max} + zS^2_{max}\right)\right) \tag{9}$$

$$= \mathcal{O}\left(|D|^3 + |D|h^3 S^3_{max} + |D|^2 S^2_{max}\right). \tag{9}$$

As complexity is particularly crucial for large datasets, i.e. large $|D|$, S_{max} can safely be considered smaller than $|D|$ leading to

$$\mathcal{O}(|D|^3 + |D|h^3 S^3_{max}). \tag{10}$$

4. Evaluation protocol

This section introduces the evaluation protocol adopted in this study; it covers technical details related to the datasets as well as the metrics that have been used.

Considering a set of documents annotated by concepts defined into a partial ordering, the main aim of the evaluation is to discuss the relevance of the clusters of documents that are automatically proposed by evaluated approaches with regard to those that would be expected by some end-users or domain experts. The dataset used to implement this evaluation protocol is based on the data made available by [48]: free terms annotations for sets of bookmarks, each of them being linked to a specific user account of the `del.icio.us` real-world bookmarking platform.[2] More particularly, the dataset we used was derived from the WordNet-based disambiguated version of the aforementioned bookmark annotations [49] — WordNet is an established and largely used knowledge organization providing a partial ordering among unambiguous sets of synonyms [50]. In this version, the free terms annotating the bookmarks have been replaced by synsets defined into WordNet 3. Unfortunately this specific version of WordNet contains cycles, which makes impossible the use of most of classical semantic similarity measures that have been proposed to deal with concepts defined into partial orders. We therefore used Wordnet 3 to Wornet 3.1 mappings, i.e. correspondences, to obtain the bookmark annotations that are compatible with WordNet 3.1 — this latter version defining a (cycle-free) partial order, which is compatible with all semantic similarity measures. Technical aspects related to the mapping are provided in the documentation associated to the dataset that can be used to reproduce the experiments.[3] Because of these modifications, the dataset we obtain provides a set of 591 documents (i.e. bookmarks) annotated by unambiguous concept-alike objects defined into a partial order. The number of annotations per document ranges from one to seven with an average of 2.67 annotations per bookmark. Fig. 2 shows the distribution of annotations of the bookmarks composing the collection.

The final part of the evaluation dataset construction consists of obtaining the expected clusters of documents that will be used to criticize the hierarchical clusters automatically generated by evaluated approaches. To our knowledge, no existing dataset provides expected hierarchical clusters among documents annotated by sets of concepts only. To answer our need, we therefore developed a tool enabling end-users to define expected hierarchical clusters for sets of documents annotated by concepts defined into a partial order. This tool gives the user the possibility (i) to consult the annotations associated to a specific document, (ii) to navigate into the partial ordering of concepts, (iii) to group documents into clusters, and (iv) to define hierarchical ordering among those clusters. In the performed evaluation, the set of evaluators was composed of 12 adult users, all familiar with the intuitive notion of concept partial ordering. Independently to each other, and through a web

[2] https://delicious.com

[3] http://benchmark.nicolasfiorini.info

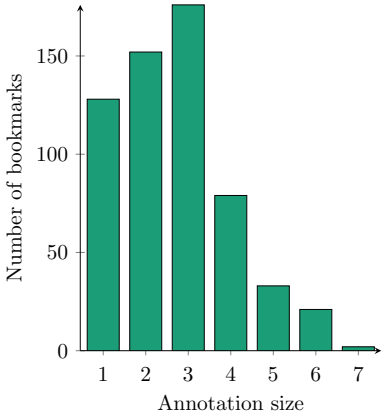

Fig. 2. Distribution of the number of annotations associated to the bookmarks of the whole collection (D1,... ,D8).

graphical interface, they have been asked to organize several collections of documents. For each collection, an evaluator had to define a hierarchical clustering of the set of documents by considering the annotations associated to the documents. In order not to introduce any bias into the experiment, no definition of expected results other than the following has been provided: "build a hierarchical clustering in order to manage the collection of proposed documents with regard to their topics (as defined by their annotations)". Prior to the data acquisition, all evaluators have been introduced to the tool and to the task by participating to an introductory session explaining the aim of the experiment as well as the functionalities provided by the tool. It is however important to stress that the evaluators were asked not to reconsider the annotations of the documents. Webpages have been anonymized, so they could not take into account additional information obtained by consulting the webpages corresponding to the bookmarks– the aim being to evaluate human vs machine hierarchical clusters that are obtained with the same input data. Each evaluator has provided one to three hierarchical clusters, each of them corresponding to a set of about 70 randomly selected bookmarks among the collection introduced above. Fig. 3 shows the number of annotations associated to the bookmarks of the seven datasets built from the original collection — these datasets are used for the evaluation. We note that on average a bookmark has two annotations with a maximal number of seven annotations.

Hierarchical clusters have been compared using the Robinson-Foulds distance [51] that calculates the topological differences of two (non-)binary trees. The output is not normalized and represents the number of basic operations needed to transform a tree into the other one. In order to normalize it, we use the statement from Pattengale et al. [52] that an unrooted tree of n leaves induces at most $2n-3$ clusters. As all datasets have a variable number of documents, the similarity of two trees t_1, t_2 is thus estimated as $TreeSim(t_1, t_2) = \frac{sim_{RF}(t_1,t_2)}{len(t_1)+len(t_2)-3}$, where $sim_{RF}(t_1, t_2)$ is the Robinson-Foulds distance of t_1, t_2 and $len(t_1) = len(t_2)$ is the number of leaves (documents) in the trees.

Analyses related to evaluator inter-agreement, i.e. dispersion among the clusters they have defined, have been made on the set of evaluators who have been asked to organize the same 7 sets of documents. To this end, for each dataset, we have studied the average distance between two hierarchical clusters as a way to objectively measure expert agreement. With an average distance of 0.159 between hierarchical clusters, the results clearly highlight the fact that an agreement between proposed expected document organizations exists — details of these results are discussed in the next section.

Table 1
Evaluation of clustering results. The average distance of expert trees with each other and of LSC, HSC and the baseline (including the post process) with the expert trees for each dataset ($D1 \ldots D7$). Lower values are better.

	D1	*D2*	*D3*	*D4*	*D5*	*D6*	*D7*
expert	0.138	0.156	0.168	0.197	0.159	0.131	0.166
LSC	**0.186**	0.240	0.240	**0.208**	**0.206**	0.271	0.210
HSC	**0.186**	**0.216**	**0.215**	0.225	**0.206**	**0.244**	**0.207**
baseline	0.269	0.274	0.267	0.283	0.288	0.326	0.276

Table 2
Evaluation of clustering computation times. The average running time (in ms) of the different clustering approaches (LSC, HSC, baseline), for each dataset ($D1 \ldots D7$). Lower values are better.

	D1	*D2*	*D3*	*D4*	*D5*	*D6*	*D7*
LSC	846	927	1073	1018	908	967	946
HSC	4680	4772	4749	5131	5424	4342	4670
baseline	**595**	**622**	**564**	**485**	**485**	**482**	**487**

5. Results

Let us compare our two approaches called Light Semantic Clustering (LSC) and Heavy Semantic Clustering (HSC), with a classical HAC combined with a naive cluster annotation (further referred to as the baseline approach). The baseline clustering approach starts with a matrix of pairwise document semantic similarities (the same that is used for LSC and HSC) but it updates this matrix using the average linkage criterion instead of a semantic similarity measure between node labels. The other difference with LSC and HSC is that in this baseline approach, each cluster is annotated by all labels annotating at least one of the documents present in this cluster (further refer to as the `merge` annotation approach).

5.1. Evaluation of the clustering approach

Expert trees differ from one another, so by calculating for each dataset the average distance between two expert trees we obtain a measure of expert inter-agreement. Ideally, we would like automatically generated cluster trees to have an average distance to expert trees lower than or close to this inter-agreement value, which would indicate that the automatic solution is as consistent with experts as experts are with one another. We compare cluster trees using the normalized Robinson-Foulds distance that reflects the percentage of clusters present in one tree but not in the other. Therefore the lower the distance, the higher the resemblance. Table 1 summarizes the average distance among expert trees and the average distance of LSC, HSC and the baseline trees with the expert ones for each dataset. It appears that both LSC and HSC clearly outperform the baseline on all datasets. Note that LSC has perfomances close to that of HSC. Indeed, LSC is as good as HSC for datasets *D*1 and *D*5 ; slightly better for dataset *D*4 (0.208 vs 0.225) but slightly less good for *D*2 (0.240 vs 0.216), *D*3 (0.240 vs 0.215), *D*6 (0.271 vs 0.244) and D7 (0.201 vs 0.27).

We also compare the computation times that have been obtained by running each method 100 times for each dataset, as shown in Table 2. As expected, the baseline is faster than our more elaborated approaches since the node labeling step is trivial. The computation times achieved by LSC are still reasonable as they are about twice those of the baseline whereas HSC is often more than 10 times slower.

As LSC provides results almost as good as those of HSC while being much faster, hence more scalable, we favor LSC over HSC and from now on we will only compare LSC to the baseline to keep the text clear.

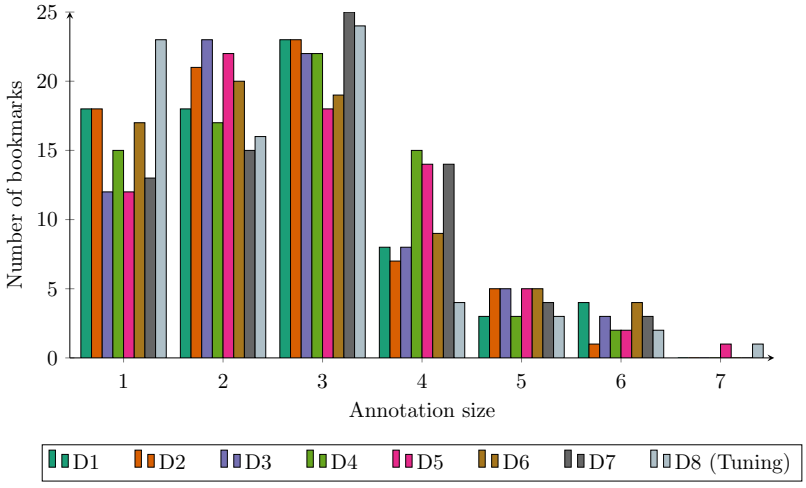

Fig. 3. Distribution of the number of annotations associated to the bookmarks of each dataset (D1,... ,D8).

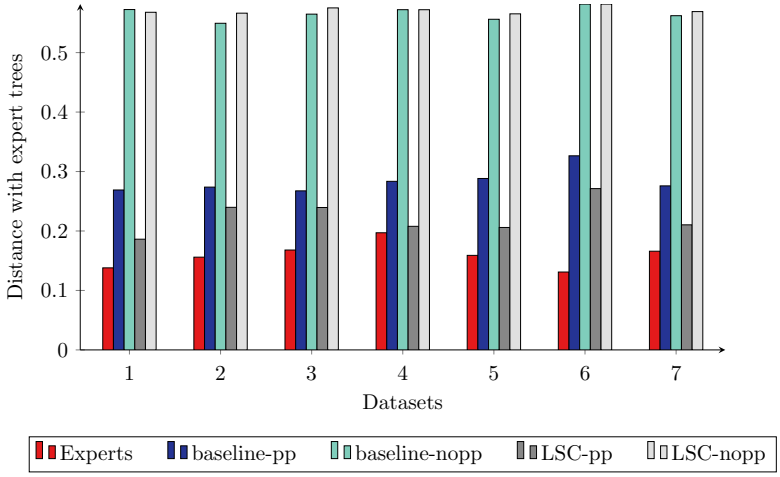

Fig. 4. Semantic clustering results. Average distances of expert trees with each other (red bars), with trees obtained using LSC (grey bars) and with trees obtained with the baseline (blue bars). The automatic clustering approaches are combined (darker color bars) or not (lighter color bars) with the flattening post-process. Lower values are better. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We conducted a more thorough study on the impact of the post-processing on LSC and the baseline. Two strategies are explored regarding the post-processing: with the flattening post-process (pp) and without it (nopp). Fig. 4 provides, for the seven datasets, the average distance to expert trees for LSC and baseline approaches with or without the flattening. The figure also depicts the expert tree inter-agreement to ease the comparison.

The results show that the flattening is an important process for both approaches. With no post-process, both methods are equally bad as they do not fit the expert expectation of a non-binary cluster tree. When the post-process is applied, LSC systematically performs better than the baseline. This emphasizes the benefit of conducting a tight imbrication of clustering and annotation, as done in LSC, so that the resulting cluster tree has a topology that is consistent with its node annotations. In fact, for some datasets (1, 2 and 7) the results of LSC (with post-processing) are congruent with the experts as they are close to the inter-agreement values that represent the experts discrepancies.

5.2. Evaluation of cluster labeling

Unfortunately, it is impossible to simultaneously evaluate the clusters along with their labels. Indeed, the experts provide different trees with different labels that can thus not be summarized in a gold standard labeled tree for each dataset. Cluster labels are thus evaluated as follows. An expert tree is provided to our method and the module — hereafter denoted by `CL` — that annotates each node in LSC is run. The result of such an approach is an index for each node of each expert tree for each dataset. The evaluation of labels follows our previous work proposal [40], in tune with other studies that suggested that semantic similarities better assess the quality of a semantic annotation [45]. For each expert tree we thus compute the average semantic similarity of our labels with the expert ones. Then, for each dataset, the scores are averaged over the expert trees hence leading to a single label score per dataset. Label scores obtained with our `CL` annotation approach and with the naive `merge` annotation approach are proposed in Fig. 5.

At first glance, the scores seem to be pretty close. In fact, they mostly depend on the datasets. The worst scores are obtained with dataset 2 for both methods. Although the scores of `merge` are similar to those of `CL` for three datasets, `CL` provides overall slightly more accurate labels — the average semantic score of `CL` is 0.494 and that of `merge` is 0.469.

The main difference between those two labelling procedures resides not so much in the accuracy of the proposed labels but in their concision. Table 3 contains the label sizes for each method. The `merge` approach contains many more concepts than the other as it does not process them. The sizes of labels for `CL` compared

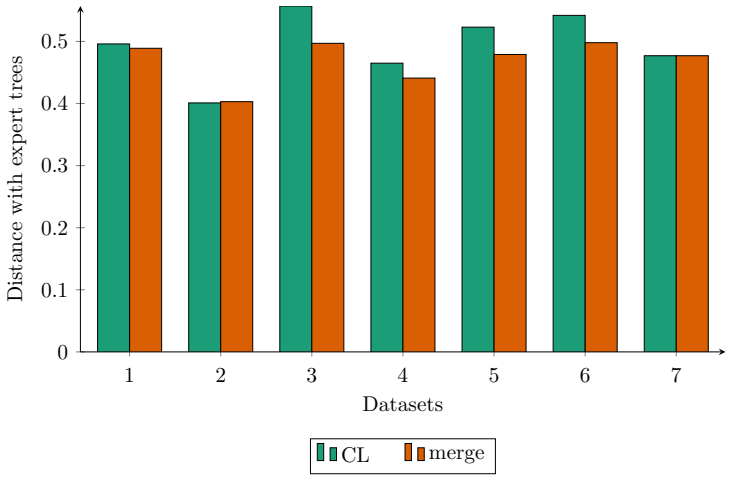

Fig. 5. Semantic cluster labeling results. Similarity of the cluster labeling proposed by "CL" and "merge" approaches with those proposed by experts. Higher values are better.

Table 3
Comparison of the average label sizes of our cluster labeling approach (CL) with the baseline and the expert data for each dataset ($D1 \ldots D7$). Lower values are better.

	$D1$	$D2$	$D3$	$D4$	$D5$	$D6$	$D7$
CL	**2.16**	**1.96**	**2.05**	**1.82**	**2.05**	**1.84**	**1.75**
merge	16.02	11.85	12.70	13.84	14.72	15.97	16.28
experts	1.11	1.05	1.03	1.08	1.13	1.12	1.06

to that of merge demonstrate that we can clearly summarize 10 to 15 concepts in 2 concepts in average without decreasing the quality of the labels — in fact, it even increases it. Note also that the number of concepts provided by CL is pretty close to that provided by the experts, compared to merge.

We observe that the scores in general are less good than for document annotation for example, where we usually get semantic scores around 0.8 [53]. The reason here is that there are outliers in each dataset that are hardly clusterizable because their annotation is vague or has nothing in common with other documents. Some examples are "corner", or "tip". Because there is no access to the corresponding webpages, it is difficult to classify these outliers. Some experts gather them in a cluster labeled VARIOUS for instance, some others strive to find a category for them. Our algorithm cannot predict the former cognitive strategy and fails at labeling the resulting "catch-all" cluster. Our cluster labeling also fails to mimic the expert labels of the latter strategy, because usually, once the expert has put the outlier in a cluster, he/she ignores it to annotate the cluster. When our algorithm annotates a cluster it can possibly ignore outliers when they are really underrepresented — because there must be enough leaves to veil them. Note that when outliers are (manually) removed from the expert trees, the semantic score of labels is of 0.78 in average, which proves their impact on our approach.

6. Conclusion

The semantic hierarchical agglomeration clustering method this paper proposes was motivated by previous works both in clustering and labeling. Although several approaches tried to make use of semantic data (GO concepts, metadata, concept mapping, etc.) there was no generic approach relying on solid solutions such as semantic similarities. When documents are annotated by groups of concepts from a taxonomy, the method we propose uses a group-wise semantic similarity measure to compute the pairwise similarities between documents, first for creating the label-based similarity matrix. When two clusters are agglomerated into a new one, a semantic annotation is automatically computed for this new cluster and then used to iteratively update the label-based similarity matrix. In this way, our clustering method guarantees the consistency between the agglomeration phase and the labeling one: semantic similarities help building more interpretable and meaningful clusters. We have proposed a benchmark containing eight datasets of about 70 bookmarks each, one for tuning the method and seven others to evaluate the approach: the hierarchical labeled clusters our method generates has been compared to trees experts proposed as a bookmark organization. Furthermore, an algorithmic effort has been made to optimize the time complexity of the proposed solution. As a result our solution has a time complexity equivalent to a naive hierarchical clustering approach (and comparable running time on our benchmarks) while providing significantly more accurate results. Such a hierarchical clustering and labeling indicates seemingly good prospects for future works concerning adaptive indexing: indexing may be context dependent (e.g. an algebra book is not labeled in the same way in a scientific library as in a municipal library; the required granularity of the annotation hangs on the use that is made of it) and our approach provides relevant hints to adaptively manage the indexing process.

To the best of our knowledge, this is the only approach that proposes a so tight imbrication of document clustering and cluster annotation. A consequence of the resulting tree topology/annotation consistence is that the cluster post-processing we propose, to filter out meaningless clusters, is more efficient and allows to provide a non-binary hierarchical organization of documents that meets human expert expectations.

All resources and results are made available at: http://sc.nicolasfiorini.info

Acknowledgment

This work was partly supported by a grant from AVieSan national program (French Alliance nationale pour les sciences de la Vie et de la Sant) and by the French Agence Nationale de la Recherche (ANR-10-BINF-01 Ancestrome).

References

[1] A. Civan, W. Jones, P. Klasnja, H. Bruce, Better to organize personal information by folders or by tags?: The devil is in the details, Proc. Am. Soc. Inf. Sci. Technol. 45 (1) (2008) 1–13, doi:10.1002/meet.2008.1450450214.
[2] O. Bergman, N. Gradovitch, J. Bar-Ilan, R. Beyth-Marom, Folder versus tag preference in personal information management, J. Am. Soc. Inf. Sci. Technol. 64 (10) (2013) 1995–2012, doi:10.1002/asi.22906.

[3] R. Xu, D. Wunsch, Survey of clustering algorithms, IEEE Trans. Neural Netw. 16 (3) (2005) 645–678.
[4] S. Harispe, S. Ranwez, S. Janaqi, J. Montmain, Semantic Similarity from Natural Language and Ontology Analysis, Synthesis Lectures on Human Language Technologies, 8, Morgan & Claypool Publishers, 2015.
[5] A. Kuhn, S. Ducasse, T. Gîrba, Semantic clustering: Identifying topics in source code, Inf. Software Technol. 49 (2007) 230–243, doi:10.1016/j.infsof.2006.10.017.
[6] P. Clerkin, P. Cunningham, C. Hayes, Ontology discovery for the semantic web using hierarchical clustering, Semantic Web Mining (2001) 27.
[7] G. Bharathi, D. Venkatesan, Study of ontology or thesaurus based document clustering and information retrieval, J. Eng. Appl. Sci. 7 (4) (2012) 342–347.
[8] A. Hotho, A. Maedche, S. Staab, Text clustering based on good aggregations, Proceedings IEEE International Conference on Data Mining, ICDM 2001 (2001) 607–608.
[9] A. Hotho, A. Maedche, S. Staab, Ontology-based text document clustering, Proceedings of KI (2002) 1–13.
[10] A. Hotho, S. Staab, G. Stumme, Wordnet improves Text Document Clustering, Third IEEE International Conference on Data Mining, ICDM 2003 (2003) 541–544.
[11] R. Baghel, D.R. Dhir, A Frequent Concepts Based Document Clustering Algorithm, Int. J. Comput. Appl. 4 (5) (2010) 6–12, doi:10.5120/826-1171.
[12] T.D. Breaux, J.W. Reed, Using ontology in hierarchical information clustering, Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS'05 (2005).
[13] J. Sedding, D. Kazakov, Wordnet-based text document clustering, proceedings of the 3rd workshop on robust methods in analysis of natural language data (2004) 104–113.
[14] G. Spanakis, G. Siolas, a. Stafylopatis, Exploiting Wikipedia Knowledge for Conceptual Hierarchical Clustering of Documents, Comput. J. 55 (3) (2011) 299–312, doi:10.1093/comjnl/bxr024.
[15] W. Song, C.H. Li, S.C. Park, Genetic algorithm for text clustering using ontology and evaluating the validity of various semantic similarity measures, Exp. Syst. Appl. 36 (5) (2009) 9095–9104, doi:10.1016/j.eswa.2008.12.046.
[16] X. Liu, W.B. Croft, P. Oh, D. Hart, Automatic recognition of reading levels from user queries, Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (2004) 548–549.
[17] B. Adryan, R. Schuh, Gene-Ontology-based clustering of gene expression data, Bioinformatics (Oxford, England) 20 (16) (2004) 2851–2, doi:10.1093/bioinformatics/bth289.
[18] J.A. Nasir, I. Varlamis, A. Karim, G. Tsatsaronis, Semantic smoothing for text clustering, Knowledge-Based Syst. 54 (2013) 216–229.
[19] A. Figueroa, G. Neumann, Context-aware semantic classification of search queries for browsing community question–answering archives, Knowledge-Based Syst. 96 (2016) 1–13.
[20] Y. Liu, M. Liu, X. Wang, Towards semantically sensitive text clustering: A feature space modeling technology based on dimension extension, PloS one 10 (3) (2015) e0117390.
[21] S. Zhu, J. Zeng, H. Mamitsuka, Enhancing medline document clustering by incorporating mesh semantic similarity, Bioinformatics 25 (15) (2009) 1944–1951.
[22] A. Maedche, V. Zacharias, Clustering ontology-based metadata in the semantic web, Principles of Data Mining and Knowledge Discovery (2002) 348–360.
[23] S. Shehata, F. Karray, M. Kamel, Enhancing Text Clustering Using Concept-based Mining Model, Sixth International Conference on Data Mining (ICDM'06) (2006) 1043–1048, doi:10.1109/ICDM.2006.64.
[24] K. Ovaska, M. Laakso, S. Hautaniemi, Fast gene ontology based clustering for microarray experiments, BioData mining 1 (1) (2008) 11, doi:10.1186/1756-0381-1-11.
[25] C.D. Manning, P. Raghavan, H. Schütze, Introduction to information retrieval, 1, Cambridge university press, 2008.
[26] F. Geraci, M. Pellegrini, M. Maggini, F. Sebastiani, Cluster Generation and Labeling for Web Snippets: A Fast, Accurate Hierarchical Solution, Internet Mathematics 3 (4) (2006) 413–443, doi:10.1080/15427951.2006.10129133.
[27] F. Role, M. Nadif, Beyond cluster labeling: Semantic interpretation of clusters' contents using a graph representation, Knowledge-Based Syst. 56 (2014) 141–155, doi:10.1016/j.knosys.2013.11.005.
[28] D. Carmel, H. Roitman, N. Zwerdling, Enhancing cluster labeling using wikipedia, Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval (2009) 139, doi:10.1145/1571941.1571967.
[29] D. Carmel, E. Yom-Tov, A. Darlow, D. Pelleg, What makes a query difficult? Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (2006) 390–397, doi:10.1145/1148170.1148238.
[30] T. Beissbarth, T.P. Speed, GOstat: find statistically overrepresented Gene Ontologies within a group of genes, Bioinf. (Oxford, England) 20 (9) (2004) 1464–5, doi:10.1093/bioinformatics/bth088.
[31] H.K. Lee, W. Braynen, K. Keshav, P. Pavlidis, ErmineJ: tool for functional analysis of gene expression data sets, BMC bioinf. 6 (2005) 269, doi:10.1186/1471-2105-6-269.
[32] S. Bauer, S. Grossmann, M. Vingron, P.N. Robinson, Ontologizer 2.0–a multifunctional tool for GO term enrichment analysis and data exploration, Bioinf. (Oxford, England) 24 (14) (2008) 1650–1, doi:10.1093/bioinformatics/btn250.
[33] J. Reimand, M. Kull, H. Peterson, J. Hansen, J. Vilo, g:Profiler–a web-based toolset for functional profiling of gene lists from large-scale experiments, Nucleic acids res. 35 (Web Server issue) (2007) W193–200, doi:10.1093/nar/gkm226.
[34] N.H. Shah, N.V. Fedoroff, CLENCH: a program for calculating Cluster ENriCHment using the Gene Ontology, Bioinf. (Oxford, England) 20 (7) (2004) 1196–7, doi:10.1093/bioinformatics/bth056.
[35] S. Maere, K. Heymans, M. Kuiper, BiNGO: a Cytoscape plugin to assess overrepresentation of gene ontology categories in biological networks, Bioinf. (Oxford, England) 21 (16) (2005) 3448–9, doi:10.1093/bioinformatics/bti551.
[36] J. Paquette, T. Tokuyasu, EGAN: exploratory gene association networks, Bioinf. (Oxford, England) 26 (2) (2010) 285–6, doi:10.1093/bioinformatics/btp656.
[37] H. Mi, Q. Dong, A. Muruganujan, P. Gaudet, S. Lewis, P.D. Thomas, PANTHER version 7: improved phylogenetic trees, orthologs and collaboration with the Gene Ontology Consortium, Nucleic acids res. 38 (Database issue) (2010) D204–10, doi:10.1093/nar/gkp1019.
[38] Y. Yang, An evaluation of Statistical Approaches to Text Categorization, Inf. retrieval 1 (1–2) (1999) 69–90.
[39] D. Trieschnigg, P. Pezik, V. Lee, F. de Jong, W. Kraaij, D. Rebholz-Schuhmann, MeSH Up: effective MeSH text classification for improved document retrieval, Bioinformatics 25 (11) (2009) 1412–1418, doi:10.1093/bioinformatics/btp249.
[40] N. Fiorini, S. Ranwez, J. Montmain, V. Ranwez, USI: a fast and accurate approach for conceptual document annotation, BMC Bioinf. 16 (1) (2015) 1–10, doi:10.1186/s12859-015-0513-4.
[41] D. Lin, An information-theoretic definition of similarity, ICML 98 (1998) 296–304.
[42] A. Schlicker, F.S. Domingues, J. Rahnenführer, T. Lengauer, A new measure for functional similarity of gene products based on Gene Ontology, BMC bioinf. 7 (1) (2006) 302, doi:10.1186/1471-2105-7-302.
[43] S. Harispe, S. Ranwez, S. Janaqi, J. Montmain, The semantic measures library and toolkit: fast computation of semantic similarity and relatedness using biomedical ontologies, Bioinformatics 30 (5) (2014) 740–2, doi:10.1093/bioinformatics/btt581.
[44] N. Fiorini, S. Ranwez, S. Harispe, J. Montmain, V. Ranwez, Usi at bioasq 2015: a semantic similarity-based approach for semantic indexing, in: Working Notes for the Conference and Labs of the Evaluation Forum (CLEF), Toulouse, France, 2015.
[45] A. Névéol, K. Zeng, O. Bodenreider, Besides precision & recall: Exploring alternative approaches to evaluating an automatic indexing tool for medline, AMIA Annu. Symp. Proc. 2006 (2006) 589.
[46] P. Resnik, Using information content to evaluate semantic similarity in a taxonomy, in: Proceedings of IJCAI-95, 1995, pp. 448–453.
[47] N. Saitou, M. Nei, The neighbor-joining method: a new method for reconstructing phylogenetic trees, Mol. biol. evol. 4 (4) (1987) 406–425.
[48] R. Wetzker, C. Zimmermann, C. Bauckhage, Analyzing social bookmarking systems: A del.icio.us cookbook, in: Proceedings of the ECAI 2008 Mining Social Data Workshop, 2008, pp. 26–30.
[49] P. Andrews, J. Pane, I. Zaihrayeu, Advanced Language Technologies for Digital Libraries: International Workshops on NLP4DL 2009, Viareggio, Italy, June 15, 2009 and AT4DL 2009, Trento, Italy, September 8, 2009, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 114–134, doi:10.1007/978-3-642-23160-5_8.
[50] G.A. Miller, Wordnet: a lexical database for english, Commun. ACM 38 (11) (1995) 39–41.
[51] D. Robinson, L.R. Foulds, Comparison of phylogenetic trees, Math. Biosci. 53 (1) (1981) 131–147.
[52] N.D. Pattengale, E.J. Gottlieb, B.M. Moret, Efficiently computing the robinson–foulds metric, J. Comput. Biol. 14 (6) (2007) 724–735.
[53] N. Fiorini, S. Ranwez, J. Montmain, V. Ranwez, Coping with imprecision during a semi-automatic conceptual indexing process, Information Processing and Management of Uncertainty in Knowledge-Based Systems proceedings (2014) 11–20.