Multiscale Visualization and Exploration of Large Bipartite Graphs

Nicola Pezzotti, Jean-Daniel Fekete, Thomas Höllt, Boudewijn Lelieveldt, Elmar Eisemann, Anna Vilanova

Eurographics Conference on Visualization (EuroVis) 2018
J. Heer, H. Leitte, and T. Ropinski
(Guest Editors)

Volume 37 (2018), Number 3

Multiscale Visualization and Exploration of Large Bipartite Graphs

Nicola Pezzotti[1], Jean-Daniel Fekete[2], Thomas Höllt[1,3], Boudewijn P.F. Lelieveldt[1,3], Elmar Eisemann[1] and Anna Vilanova[1]

[1]TU Delft, The Netherlands
[2]INRIA, Saclay, France
[3]Leiden University Medical Center, Leiden, The Netherlands

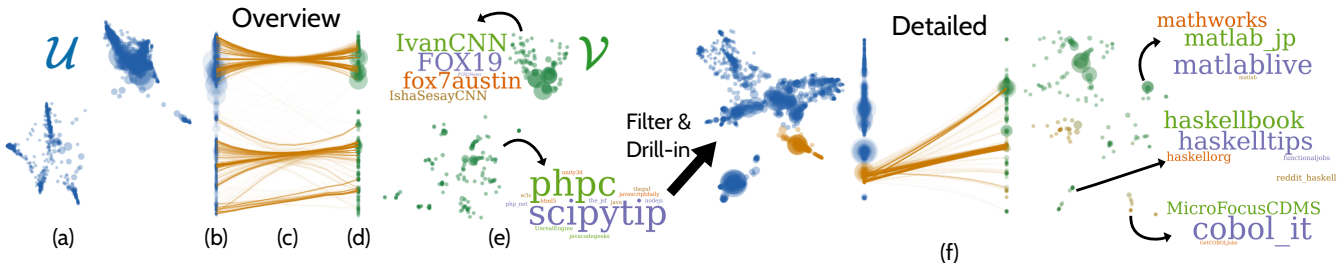

Figure 1: ***Example of WAOW-Vis*** *for a bipartite graph containing a collection $\mathcal{U}$ of 9.4M Twitter users is linked to a collection of 228 $\mathcal{V}$ of Twitter feeds associated to programming languages and US' news outlets. The bipartite graph is preprocessed by the HSNE algorithm that extracts a hierarchy of landmarks, i.e., sets of vertices. In the overview, landmarks in the highest scale for the two collections are placed in (b,d) 1- and (a,e) 2-dimensional embeddings that reveal major clusters of similarly connected vertices (c). The hierarchy is explored with an overview-first and details-on-demand approach that reveals hierarchies of sub-clusters (f).*

Abstract

A bipartite graph *is a powerful abstraction for modeling relationships between two collections. Visualizations of bipartite graphs allow users to understand the mutual relationships between the elements in the two collections, e.g., by identifying clusters of similarly connected elements. However, commonly-used visual representations do not scale for the analysis of large bipartite graphs containing tens of millions of vertices, often resorting to an a-priori clustering of the sets. To address this issue, we present the Who's-Active-On-What-Visualization (WAOW-Vis) that allows for multiscale exploration of a bipartite social-network without imposing an a-priori clustering. To this end, we propose to treat a bipartite graph as a high-dimensional space and we create the WAOW-Vis adapting the multiscale dimensionality-reduction technique HSNE. The application of HSNE for bipartite graph requires several modifications that form the contributions of this work. Given the nature of the problem, a set-based similarity is proposed. For efficient and scalable computations, we use compressed bitmaps to represent sets and we present a novel space partitioning tree to efficiently compute similarities; the Sets Intersection Tree. Finally, we validate WAOW-Vis on several datasets connecting Twitter-users and -streams in different domains: news, computer science and politics. We show how WAOW-Vis is particularly effective in identifying hierarchies of communities among social-media users.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

A Bipartite Graph is an important abstraction in computer science. Vertices in the graph are divided into two disjoint and independent collections of items $\mathcal{U}$ and $\mathcal{V}$. The edges in the graph represent the relationships between the elements in the two collections and, therefore, only connect elements in $\mathcal{U}$ to elements in $\mathcal{V}$. Several problems can be modeled as a bipartite graph analysis, for example, the two collections may be software developers and source files, gene mutations and patients in a cohort study or social media users and the news outlets they follow on social media.

Previous work [HMM00, VLKS*11] has identified the following analytical tasks for the exploratory analysis of bipartite graphs.

- **(T1)** Identifying clusters of similar elements in $\mathcal{U}$ with regard to their connections to elements in $\mathcal{V}$ and vice versa.
- **(T2)** Understanding the interrelationships between the clusters in the two collections $\mathcal{U}$ and $\mathcal{V}$.

A widely used approach for performing these tasks is to visualize the bipartite graph using a node-link visualization. A commonly used approach separates the two collections of items on screen. Vertices are displayed as points arranged along two parallel axes, i.e.,

corresponding to $\mathcal{U}$ and $\mathcal{V}$, and the edges are visualized as lines connecting the vertices [HMM00, VLKS*11], see Figure 2a. By using matrix reordering algorithms on the adjacency matrix of the graph [BBHR*16, VLKS*11], vertices that share a similar connection pattern with respect to the other collection can be placed close together along the axis, allowing for identification of vertices with similar connections **(T1)** and their mutual relationships **(T2)**. However, node-link visualizations do not scale for the analysis of bipartite graphs containing more than a few hundreds vertices due to the resulting visual clutter [GFC05]. To overcome this limitation, algorithmic graph preprocessing techniques are often used to reduce the complexity of the graph to be drawn [VLKS*11]. For bipartite graphs, biclustering algorithms, also known as co-clustering techniques, become the standard for the identification of sub-clusters in $\mathcal{U}$ and $\mathcal{V}$ that share a similar connection pattern to the other collection [HSBW11, MO04, OKHC14, PGAR15]. Clusters are then visualized as aggregated vertices in the node-link diagram. Nonetheless, such an approach assumes that there is no variability within a cluster, which is problematic when the data is large and contains a hierarchy of sub-clusters. For example, communities of social media users may share similar connections to a set of news feeds, but they may also contain sub communities where the connections are slightly different.

To address this problem we present Who's-Active-On-What-Visualization (WAOW-Vis), a technique designed to reveal hierarchies of clusters in bipartite graphs. To this end, we adapt a multiscale dimensionality-reduction algorithm, the "Hierarchical Stochastic Neighbor Embedding" (HSNE) [PHL*16], for extracting and visualizing clusters of similarly connected vertices. Figure 1b-d shows the resulting layout of WAOW-Vis, where two 1-dimensional embeddings are used to visualize the vertices in a layout that mimic node-link visualizations for bipartite graphs **(T1)**. Moreover, we show that, by adding 2-dimensional embeddings of the same vertices, we obtain more detailed insight on the interrelationships between the clusters **(T2)**. As an illustration of our results and goal, in Figure 1a we can identify clusters and see the internal structure much easier than in Figure 1b. However, Figure 1a-e, shows only one scale of the hierarchy computed by HSNE. As a matter of fact, HSNE does not embed the dataset in its entirety but it selects representative vertices at different scales. i.e., landmarks. The visualization of the landmarks in the highest scale shows an *overview* of the main clusters of connected vertices in the graph. The user can then select these clusters and ask for a *detailed visualization* that reveals more sub-clusters for the landmarks in the lower scale, as shown in Figure 1f.

Several challenges need to be overcome to achieve our goals using HSNE. Dimensionality reduction algorithms rely on a dense representation of the data for the computation of the similarities between points. This fact constitutes a problem if the dimensionality-reduction is applied to a large adjacency matrix that is saved in a dense format, as it will not fit in memory. Consider for example a dataset that we mined from Twitter, which we will present in Section 7. It comprises of 19M users linked to 329 user interests. The dense representation of the biadjacency matrix would occupy 23 GB in memory if organized as required by dimensionality-reduction algorithms such as HSNE. We propose a novel computational pipeline that addresses the memory and computational scal-

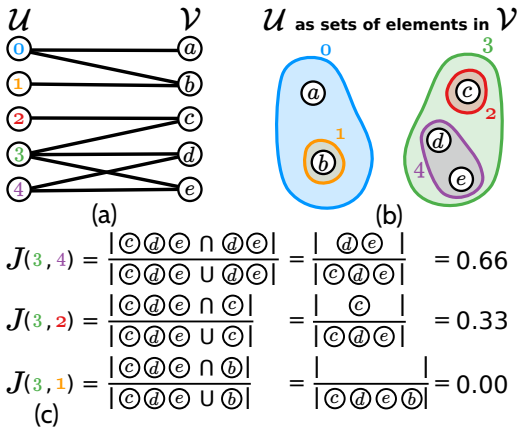

Figure 2: ***Computation of the similarities*** *between the vertices in $\mathcal{U}$. Vertices in $\mathcal{U}$ are seen as sets of elements in $\mathcal{V}$ (b). The Jaccard similarities are computed as the cardinality of the intersection divided by the cardinality of the union (c).*

ability problem for undirected and non-weighted bipartite graphs. Here, we propose to treat the data as a collection of sets. Then we adopt the Jaccard similarities, a compelling metric that measure the amount of shared links, to compute similarities between vertices. Figure 2 shows an example of how the similarities for the collection $\mathcal{U}$ are computed. The vertices in $\mathcal{U}$ are seen as sets of elements in $\mathcal{V}$. The Jaccard similarities are computed as the number of shared elements between sets divided by the size of their union. A similar computation is done for the vertices in $\mathcal{V}$. Scalable computations and memory usage are enabled by the use of compressed bitmaps [LSYKK16] to represent the sets and by a novel tree-based data structure, the Sets Intersection Tree (SIT), to efficiently compute the Jaccard similarities. Moreover, we show that by combining the k-nearest neighborhood graph, computed as input for HSNE, the resulting embeddings reveal clusters of vertices at different scales **(T1)** and their interrelationships **(T2)**. Finally, since the user can be confused by the fact that same vertices may appear in different locations in the 1- and 2-dimensional embeddings, we present a technique that enforces similar positions on the vertical axis for the same vertices to facilitate the creation of a mental map.

The contribution of this work is the WAOW-Vis framework, that allows for analyzing bipartite graphs on a social-network scale at different levels of detail. The development of WAOW-Vis is made possible by a set of additional contributions of this paper:

- The usage of compressed bitmaps as high-dimensional data representation for bipartite-graphs.
- A data structure for the efficient computations of similarities between compressed bitmaps in the Jaccard space; the Sets Intersection Tree.
- A modification of the tSNE [vdMH08] algorithm for the creation of consistent and interrelated embeddings.

The rest of this paper is organized as follows. In the next section we present the related work. An overview of WAOW-Vis is given in Section 3. Section 4 highlights the generation of the hierarchical representation of the data. In Section 5 we present the

interactive exploration using WAOW-Vis. Finally, in Section 7 we validate WAOW-Vis via several test cases based on three datasets that we obtained by mining Twitter. We demonstrate how WAOW-Vis is particularly effective in highlighting communities of users that shares similar interests.

2. Related Work

A **Bipartite graph** $(\mathcal{U},\mathcal{V},E)$ is a particular type of multifaceted graph [HSS15] where vertices form two disjoint sets $\mathcal{U}$ and $\mathcal{V}$, and edges $E = (u,v) \in \mathcal{U} \times \mathcal{V}$ connect elements from each sets. Bipartite graphs can be displayed using traditional visual encodings for graphs such as **node-link diagrams** and **adjacency matrices** [VLKS*11]. In a node-link diagram vertices are placed in a 2-dimensional space using a layout algorithm [Tam07]. Adjacency-matrix visualizations rely on matrix reordering techniques for highlighting connectivity patterns, e.g., cliques of strongly connected nodes [BBHR*16, VLKS*11]. Empirical studies show that node-link visualizations are usually more intuitive for understanding the graph but, for dense graphs, adjacency matrix visualizations outperform node-link visualizations due to the reduced visual clutter [GFC05]. Node-link diagrams relying on dimensionality reduction techniques have also been presented [MAH*12], but they do not scale beyond few thousand vertices. To combine the advantages of both worlds, **hybrid techniques** had been developed. The MatrixExplorer [HF06] and the NodeTrix [HFM07] visualization systems are just two examples that combine node-links and adjacency matrix visualization in order to provide greater insights. WAOW-Vis is a hybrid technique that scales to bipartite-graphs with several millions vertices and edges. Similarly to matrix reordering techniques, it allows for identifying clusters of similarly connected vertices as clusters of points in the dimensionality reduction layouts **(T1)**, i.e., embeddings (see Figure 1a). Then, thanks to a visual design similar to node-link diagrams, WAOW-Vis provides a better interpretability of the results by showing 1-dimensional embeddings and their interrelationships **(T2)** (see Figure 1b-d).

In order to enforce a distinction between the collections $\mathcal{U}$ and $\mathcal{V}$, different approaches allocate **separated visual spaces** to the two collections, which can be parallel axes, interleaved axes [BDF*10], or concentric circles [DRM12]. Edges connecting the vertices in the two collections are then visualized as links. A subset of the links may be drawn based on the user's current focus or, to give a complete picture of the data, all the links can be drawn. Lines may be bundled in order to reduce the resulting visual clutter [ZXYQ13]. This approach is used in several visual analytics systems such as VisLink [CC07], PivotPath [DRRD12], PNLBs [GKL*13] and Jigsaw [SGL08].

In this work we focus on the analysis of large graphs, i.e., containing tens of millions of vertices and edges. Gaining insight from a direct visualization of the data with one of the previously described techniques is not possible due to the resulting visual clutter. Algorithmic graph preprocessing [VLKS*11] is therefore of major importance in order to create meaningful visualizations. **Graph filtering** algorithms are used for reducing the number of the visualized vertices [BGW03, LF06]. Jia et al. [JHGH08], for example, remove vertices that are not considered important according to a notion of graph centrality. In **graph aggregation** techniques, the

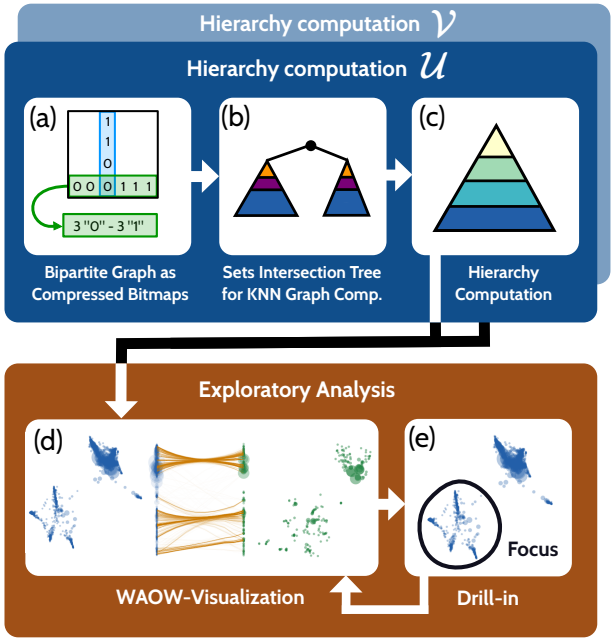

Figure 3: ***Overview*** *of WAOW-Vis. Two hierarchical representations of the collections $\mathcal{U}$ and $\mathcal{V}$ are computed. The elements in the collections are encoded via compressed bitmaps (a). The k-nearest neighborhood graph are computed for the non-duplicated bitmaps (b) and the hierarchy is computed by HSNE (c). In WAOW-Vis, we first present an overview of the data (d). The user may focus on specific regions of interest and generate more detailed visualizations (e).*

vertices are not simply removed, but multiple vertices are merged into a single one, hence reducing the size of the resulting graph. Vertices can be merged according to different criteria, e.g., by treating cliques of strongly connected vertices as a single node in the visualization [BBD*11, EDG*08]. For bipartite graphs, aggregation is usually performed using biclustering algorithms [MO04] which create clusters in the two collections $\mathcal{U}$ and $\mathcal{V}$ based on their mutual relationships. Biclustering algorithms are often used in bioinformatic [HSBW11, OKHC14, PGAR15] and for the analysis of deep neural networks [MCZ*17]. If the graph simplification is repeated multiple times, a hierarchical graph, also called **compound graph**, is created. The hierarchical graph is then analyzed with visualizations that allow for the exploration of the data at different scales [HF06,HFM07,Hol06,XCQS16]. Existing techniques, however, are limited in the analysis of large graphs. The preprocessing represents a bottleneck that requires many hours, or even days, to complete, hence limiting the interactive analysis of the data. Our goal is to tackle bipartite graphs with tens of millions of vertices and edges at interactive speed on regular hardware.

3. WAOW-Vis overview

Figure 3 shows the overview of the generation of WAOW-Vis and its interactive exploration. WAOW-Vis uses separated visual spaces for visualizing the two collections, and creates a layout similar to the traditional and easy to interpret node-link visualization for bipartite graphs, as shown in Figure 1b-d. However, we propose to

enrich the visualization with 2-dimensional embeddings, see Figure 1a and e. This solution reveals more clusters and richer information about their interrelationships, as vertices have more visual space to be layed in **(T2)**. We considered adding links between the 2D and 1D embeddings to make the identification of the same vertices easier. However, we quickly realized it would introduce excessive clutter and, instead, we developed a novel embedding computation technique keeps corresponding vertices roughly aligned (Section 5.2).

The main contribution of WAOW-Vis is it scalability to graphs that, to the best of our knowledge, cannot be handled by existing techniques. This result is achieved thanks to a novel algorithmic graph preprocessing that take advantage of the recent advancements in the field of large high-dimensional data analysis [PHL*16, TLZM16]. Our approach can be separated in two steps, the **hierarchy computation** and the **exploratory analysis** of the so computed hierarchies of vertices.

In the **hierarchy computation** step, two identical computational pipelines are applied to the collections $\mathcal{U}$ and $\mathcal{V}$ separately (Figure 3a-c). First, the biadjacency matrices of the bipartite graph are transformed into two collections of compressed bitmaps. A bitmap associated with a vertex in $u \in \mathcal{U}$ contains the set of vertices in $\mathcal{V}$ that are connected to u and vice versa (Section 4.1). Then, the compressed bitmaps are organized in a novel data structure, the Set Intersection Tree (SIT), presented in Section 4.2. The SIT allows for the efficient computation of Jaccard similarities between vertices, leading to a scalable approach for the creation of the k-nearest neighborhoods graph. The k-nearest neighborhoods graph is then used as the input for the Hierarchical Stochastic Neighbor Embedding (HSNE) [PHL*16] algorithm that generates a hierarchical representation of the collections $\mathcal{U}$ and $\mathcal{V}$ (Section 4.3). Intuitively, the resulting hierarchies contain a number of scales. Each scale contains a number of **landmarks** that can be seen as a collection of elements in the two collection of vertices $\mathcal{U}$ and $\mathcal{V}$.

The **exploratory analysis** starts from an overview visualization, an example of which is presented in Figure 1a-e. The visualization contains embeddings of the landmarks at the highest scale of the HSNE hierarchy for each collection $\mathcal{U}$ and $\mathcal{V}$ that reveal similarly connected clusters of vertices (Sections 5.1 and 5.2). Furthermore, the user can request more detailed visualizations by filtering uninteresting clusters of vertices and by drilling into the hierarchies (Section 5.3), hence generating novel layouts containing data points from a more detailed HSNE scale (Figure 1f). With this approach, the user can reveal more heterogeneity within one of the previously identified cluster of vertices **(T1)**. WAOW-Vis is designed to explore the structure of bipartite graphs and the relations between the structures of the two sets of vertices. From the perspective of the task taxonomy of graph visualization by Lee et al. [LPS*06], WAOW-Vis supports group-level and cluster-level tasks, but not path-level tasks. The node-level and link-level tasks can be supported with varying levels or precision depending on the visualized scale.

4. Hierarchy computation

In this section we present how the bipartite graph is transformed in the two hierarchical data representations that are used for the creation of WAOW-Vis. In each subsection we present a single module of computational pipeline that is introduced in Figure 3a-c.

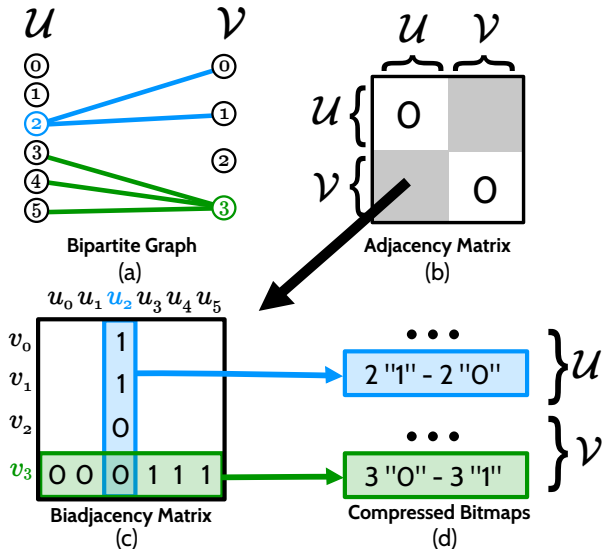

Figure 4: ***Bipartite graphs as high-dimensional data****. A bipartite graph connects a collection of users $\mathcal{U}$ with a collection of Twitter feeds $\mathcal{V}$. To avoid clutter, here we show only the links connecting u_2 and v_3 (a). The bipartite graph is represented by an adjacency matrix (b). The adjacency matrix contains two symmetric biadjacency matrices (c). The elements in $\mathcal{U}$ and $\mathcal{V}$ are seen as two high-dimensional datasets. To reduce the memory occupation and speed up the similarities computation, the data points are saved as compressed bitmaps using, for example, a Run-Length Encoding [RC67](d).*

4.1. Compressed bitmaps as high-dimensional data

Figure 4 shows an example of how the data is transformed in a collection of compressed bitmaps that are used for the efficient computation of the similarities, both in terms of memory and computations. The corresponding adjacency matrix for this data is presented in Figure 4b. The matrix can be seen as a composition of two disjoint and symmetric regions called biadjacency matrices, one of which is shown in Figure 4c. Biadjacency matrices encode the relationships between the set $\mathcal{U}$ and $\mathcal{V}$ and vice versa.

We propose to treat the biadjacency matrices as high-dimensional datasets and to measure similarities between the vertices using the Jaccard similarities as introduced in Figure 2. Given two indices i and j, the Jaccard similarity of the corresponding vertices is defined as follows:

$$J\left(\mathbf{v}_i, \mathbf{v}_j\right) = \frac{\sum_b \left(\mathbf{v}_i[b] \wedge \mathbf{v}_j[b]\right)}{\sum_b \left(\mathbf{v}_i[b] \vee \mathbf{v}_j[b]\right)}, \tag{1}$$

where $\mathbf{v}_i[b]$ is the b-th element in the i-th row of the biadjacency matrix where $\mathbf{v}_i, \mathbf{v}_j \in \mathcal{U}$. The numerator computes the number of shared elements in $\mathcal{U}$ for $\mathbf{v}_i$ and $\mathbf{v}_j$, while the denominator counts the number of elements in the union of the two.

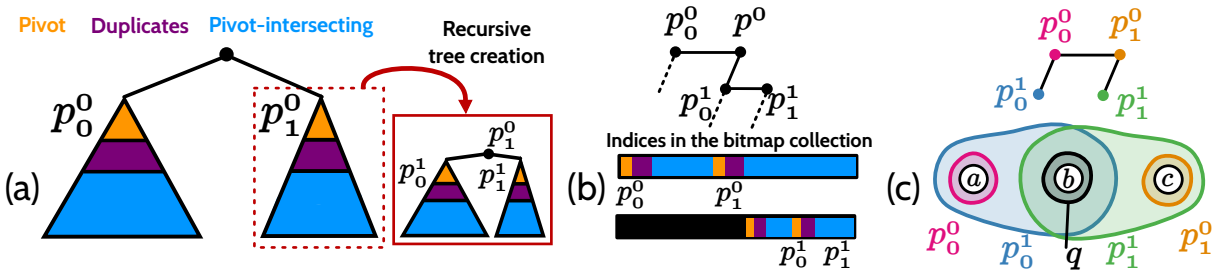

Figure 5: ***Sets Intersection Tree*** *Bitmaps are organized in a number of subtrees. (a) Every tree contains bitmaps that are intersecting with the pivot, p_0^0 and p_1^0 in this case. Every sub tree is recursively divided in subtrees. (b) The SIT is implemented with a left-child right-sibling binary-tree. Bitmaps are not actually inserted in the tree but every node references to a linear array of indices. (c) The query for a set q starts from the root, i.e., p_0^0. All the siblings of a visited node are traversed, i.e., p_1^0. Children of a node are visited if the union of the sets in the subtree are intersecting q. Both p_0^1 and p_0^0 are visited in the example.*

Dimensionality-reduction techniques requires the data to be in the form of dense matrices, i.e., losing the advantage of a sparse representation of the biadjacency matrices. Moreover, the resulting data matrix will not fit in memory for large graphs, e.g., containing tens of millions of vertices. To overcome this limitation, we propose to treat the bipartite graph as a collection of compressed bitmaps, also called bitsets, where every row in the biadjacency matrices is saved as a compressed bitmap. A bitmap is a data representation in which every element in the set is represented by a bit. Bitmaps permit the extremely fast computation of the Jaccard similarity, see Equation 1. The numerator is computed with a bitwise-AND between the two bitmaps and the denominator with a bitwise-OR. However, the memory occupation of the bitmap corresponds to the maximum number of elements in the set, i.e., the number of columns in our setting, making it identical to a dense representation. Compression techniques are used to address this problem and to dramatically reduce the memory occupation of data. A straightforward approach is to use run-length encoding (RLE) [RC67], in which repetitions of consecutive bits are stored as a single value as well as the number of times it occurs. In this way, the biadjacency matrices are transformed in two collections of compressed bitmaps representing the vertices, see Figure 4d. In WAOW-Vis we use the Roaring Bitmaps [LSYKK16], which are a hybrid data structure that combines different compression schemes on chunks of the bitmap based on their characteristics, e.g. their sparsity. Roaring Bitmaps are up to two orders of magnitude faster than traditional set implementations and are used by several Big-Data processing engines such as Apache-Lucene [MHG10] and -Spark [ZXW*16]. In the next section we present how the k-nearest neighborhood graph, which is needed for the computation of the HSNE hierarchies, is built from a large collection of compressed bitmaps.

4.2. Sets Intersection Tree

Once the biadjacency matrices are converted into two collections of bitmaps, $\mathcal{U}$ and $\mathcal{V}$, we compute the k-nearest neighborhood graph for each collection. As the procedure is the same for both, we will concentrate on the case of $\mathcal{U}$. To the best of our knowledge, no data structure exists to efficiently compute the k-nearest neighbors among compressed bitmaps. To address this problem we propose a novel tree-based data structure; the Sets Intersection Tree (SIT). Each node in this tree represents an element in $\mathcal{U}$ and the SIT will support an efficient algorithm to calculate the k-nearest neighbors of a given query element q, represented by its bitmap, by using a special traversal algorithm. The efficiency of this traversal results from the possibility for an early termination, which enables us to skip testing many elements in $\mathcal{U}$. The early termination will be enabled by two criteria. First, each node contains a union of all bitmaps of its subtree, enabling a quick test to determine if q shares any common element with any node in the subtree by using a bitwise AND operation. Second, the special construction of the tree will enable us to evaluate a bound on the Jaccard similarities of all elements in a subtree using only the bitmap of this subtree's root node, the *pivot*. Before describing the traversal algorithm, we first detail the construction, as it will facilitate deriving the bound on the Jaccard similarity.

The actual **construction** of the tree works as follows. We select the element u with the lowest cardinality to be the root node of our binary tree. Its bitmap will be used as a pivot, hence the name, to partition the remaining elements in $\mathcal{U}$, into a set $\mathcal{U}_1$, which contains all elements intersecting u (an AND operation between the bitmaps will not result in a zero), and the rest $\mathcal{U}_2$. The set $\mathcal{U}_1$ will form the left, $\mathcal{U}_2$ the right subtree of u. The subtrees are build up recursively in the same manner, choosing a pivot of lowest cardinality and building the subtrees. A special case are identical elements, which do not appear multiple times in the tree, instead each node will contain the indices of the corresponding elements in $\mathcal{U}$. Additionally, we compute the union of the bitmaps in each subtree, which will be used for the early termination. We use a bottom-up method by performing an OR operation between the bitmaps of the children of each node.

The **querying** of k-nearest neighborhoods in the SIT works as follows. Given $q \in \mathcal{U}$ for which we want to find the k-nearest neighbors, we start a recursive visit from the root of the SIT, p_0^0 in Figure 5. During the traversal, we maintain a min-heap data structure of size k that keeps track of the closest neighbors found so far; each visited node is compared against the minimal element in the heap and replaces it if its Jaccard similarity (Equation 1; the intersection divided by the union) is larger. At the end of the traversal, the heap will contain the k-nearest neighbors.

For each node, we test q's bitmap against the precomputed union of the sets in its subtree M_0^0.If $J\left(q, M_0^0\right) \neq 0$, then the traversal

continues with the children, otherwise, they have no overlap and cannot be similar (Jaccard similarity is zero). It is insufficient to test only against the bitmap of a node, as illustrated in Figure 5c; p_0^0 is not intersecting q (Jaccard similarity is 0), however, q intersects p_0^1 in the subtree, which has $J\left(q, p_0^1\right) = 0.5$.

An additional **early termination** criterion stems from the way that the SIT is constructed. By selecting the smallest set to be the pivot of a subtree, we are inherently introducing an ordering for the sets, i.e., the deeper a pivot, the larger it is. Because the denominator of the Jaccard similarity contains the union of the two sets, we can compute an upper bound for the similarities that we may find in a given sub tree. If the upper bound is lower than minimal element in the heap, we can avoid visiting the subtree.

Finally, for high efficiency, we propose a few **optimizations**, which we detail here. First, our tree does not actually store the bitmaps in the nodes, as this would lead to many copy operations of the data during the construction process. Instead, each node contains pointers to a large linear array of indices that contains all bitmaps in sequential order. The partitioning is then only affecting the indices, but not the bitmaps. Second, the construction of a subtree is stopped when only a few elements (typically 20) are left, as then the traversal cost actually exceeds the cost for testing all elements individually. This strategy follows bucket KD-Trees [ML14], where the final elements are stored in a list. This solution, together with the efficient computation of bitwise-AND and -OR granted by the RoaringBitmaps, enables us to compute the k-nearest neighborhood graphs containing several millions of nodes.

4.3. Hierarchical representation

The hierarchical representation of the biparite graph is generated by computing the Hierarchical Stochastic Neighbor Embedding (HSNE) [PHL*16]. We differ from the original HSNE algorithm , which is openly available as parte of the HDI library [Pez17], as we compute the hierarchies starting from the k-nearest neighborhoods graph computed using the Jaccard similarities (Equation 1). This HSNE result allows us to create visual clusters of vertices in $\mathcal{U}$ that share connections to the other collection $\mathcal{V}$ in a multiscale approach (**T1**). Furthermore, by combining compressed bitmaps, the SIT tree, and the HSNE algorithm, we are able to scale the computation to extremely large biadjacency matrices, making it possible to analyze dataset of a social-network scale.

More specifically, HSNE organizes the high-dimensional points or, in our case, the vertices, in a number of scales that are organized hierarchically. Each scale contains a number of landmarks that represent the complete data at the level of detail identified by the scale. Intuitively, a landmark is a collection of similarly connected vertices, where the degree of similarity is given by the position in the hierarchy. For lower scales, only vertices that shares very similar connections belong to the same landmark, while this constraint is relaxed the higher the scale in the hierarchy.

We denote the set of landmarks extracted from $\mathcal{U}$ at scale s as the collection $\mathcal{U}^s$. $\mathcal{U}^1$ represents the first scale, which is the input dataset $\mathcal{U}$. Higher scales are always subsets of previous scales, hence $\mathcal{U}^s \subset \mathcal{U}^{s-1}$. Inside a scale, the similarity between the landmarks is encoded by a transition matrix $T_{\mathcal{U}}^s$. For the first scale, $T_{\mathcal{U}}^1$ is given by the k-nearest neighborhood graph that is weighted by the similarities. Landmarks in the next scale are selected among those that have higher centrality in the graph. The centrality is computed by using the transition matrix $T_{\mathcal{U}}^1$ as a Markov Chain and by computing its stationary distribution with a Monte Carlo approach [Gey11]. Vertices with value in the stationary distribution higher than a given threshold are selected to be landmarks in the higher scale $\mathcal{U}^2$.

A link between landmarks $\mathcal{U}^2$ to the landmarks in the lower scale $\mathcal{U}^1$ is then computed. More generally, it is defined as *area of influence* of $\mathcal{U}^s$ over $\mathcal{U}^{s-1}$ and is encoded in the matrix $I_{\mathcal{U}}^s$. $I_{\mathcal{U}}^s$ has size $|\mathcal{U}^{s-1}| \times |\mathcal{U}^s|$, where $I^s(i,j)_{\mathcal{U}}$ is the probability that the landmark $\mathcal{U}_i^{s-1}$ in the previous scale is well represented, i.e., close in the k-nearest neighborhood graph, by $\mathcal{U}_j^s$. The similarity matrix $T_{\mathcal{U}}^s$ for landmarks in the new scale s encodes the overlap of the area of influence of the landmarks $\mathcal{U}^s$. The process is iterated until only a limited number of landmarks, i.e., less then a thousand, remain in the highest scale. A similar hierarchical representation is derived for the collection $\mathcal{V}$. In the next section we present how the hierarchy is visualized and explored.

5. Exploratory analysis

In this section we present the design of the visualizations used to interactively explore the hierarchical data representation of the graph. First we present the layout of a single visualization (Section 5.1). Then we present how the dimensionality reduction is performed for every collection (Section 5.2). Finally, we explain how more detailed visualizations are generated from a subset of user-selected landmarks (Section 5.3).

5.1. Visual design

Figure 1 shows an instance of WAOW-Vis. The visualization consist of four embeddings. Every point in the embeddings represent a landmark in the corresponding HSNE scale. Landmarks are placed close together if they are similar according to their Jaccard similarities at the given scale. This allows us to identify groups of elements in the collection $\mathcal{U}$ that have similar connections to $\mathcal{V}$, and vice versa. Moreover, a landmark corresponds to a set of vertices in the original collection, as presented in Section 4.3. The size of points in the visualization encodes the number of vertices represented by the corresponding landmark [PHL*16].

At the center of the visualization two 1-dimensional embeddings, one for $\mathcal{U}$ and one for $\mathcal{V}$, are used to create a layout that is similar to a traditional node-link diagram for bipartite graphs. We adopt this layout because it is reported that node-link diagrams are more intuitive for understanding the graph [GFC05]. By brushing on one of the embeddings, the vertices are selected and the links to the other collection are visualized as lines. These lines are then bundled with a real-time implementation of the force-directed bundling algorithm proposed by Holten and van Wijk [HVW09]. Each collection $\mathcal{U}$ and $\mathcal{V}$ is also shown in a 2-dimensional embedding, as shown in Figure 1a and e. The rationale behind this choice is that, the more visual space is available for the landmarks, the more interrelationships between the clusters become apparent as it can be seen by comparing Figure 1a to Figure 1b **(T2)**.

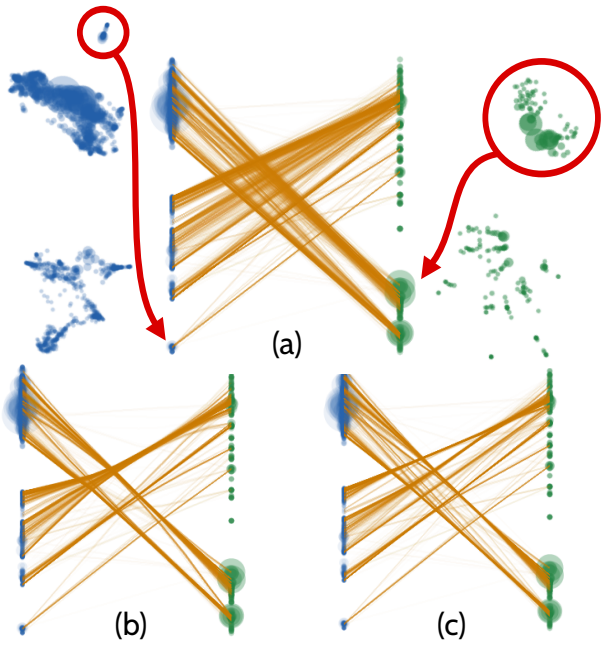

Figure 6: ***Possible problems*** *that arise if the embeddings are generated independently from each other. Here, the same data presented in Figure 1a-e is embedded with a tSNE minimization [vdMH08] instead of our approach as presented in Section 5.2. Elements in $\mathcal{U}$ and $\mathcal{V}$ are badly aligned, hence cluttering the visualization of the links between the two collections. Moreover, the same cluster of landmarks may appear in different positions along the vertical axis (a). Bundling the lines reduces clutter but does not produce a neat layout as in Figure 1a-e (b,c).*

Finally, we found that, if attributes are available for the element in the collection, it is useful for the understanding of the interrelationships between the clusters to add this information using, e.g., a word-cloud visualization. This feature leads to the identification and labeling of clusters of landmarks. In the WAOW-Vis presented in Figure 1, the two clusters in the collection $\mathcal{V}$ consists of Twitter feeds associated to two different domains, i.e., computer science and United States' news outlets.

5.2. Embedding computation and alignment

Without any additional constraint, the same clusters in the 1- and 2-dimensional embeddings might be placed in different positions along the vertical axis, which makes associations between the related embeddings a difficult task. Example of possible problems are shown in Figure 6 for the same data presented in Figure 1a-e. The two clusters highlighted in red are in different positions along the vertical axis. Furthermore, the two collections are not properly aligned, hence creating a cluttered visualization of the links. By bundling the lines, as shown in Figure 6b-c, the problem is mitigated but it is not removed. To address this issue, we implemented a modified version of the tSNE algorithm [vdMH08] enforces similar positions on the vertical axis for all landmarks of the same collection and for similarly connected landmarks in $\mathcal{U}$ and $\mathcal{V}$.

A single embedding is computed by randomly placing the landmarks in a 1- or 2-dimensional space. With an iterative gradient-descent minimization, landmarks are then moved in the embeddings in such a way that, after a number of iterations, they are close to similar landmarks according to the transition matrix $T^s_{\mathcal{U}}$. In this way, clusters of landmarks in the embedding represent groups of similar elements. More specifically, we minimize the original tSNE's cost function C_e^{tSNE} to generate the embedding e which is defined as follows:

$$C_e^{\text{tSNE}} = KL(T^s_{\mathcal{U}}||Q_e), \tag{2}$$

where $KL(T^s_{\mathcal{U}}||Q_e)$ is the Kullback-Leibler divergence between the joint-probability distributions defined by the transition matrix $T^s_{\mathcal{U}}$ and Q_e. Q_e is a joint-probability distribution that is obtained by weighting the distances between the landmarks in the embedding e with the Student's t-distribution [vdMH08]. The points are then iteratively moved in the embedding along the negative gradient of the cost function C_e^{tSNE}, until their positions reflect the similarities in $T^s_{\mathcal{U}}$. We refer the interested reader to the work of van de Maaten et al. [vdMH08] for the detail on how Q_e is computed and how the gradient-descent parameters for C_e^{tSNE} are chosen.

In order to take into consideration the position of landmarks in a set of embeddings F, hence enforcing the alignment between the same landmarks, we modify the cost function C_e as follows:

$$\begin{aligned} C_e &= (1-\alpha)\, C_e^{\text{tSNE}} &&+\alpha C_e^{\text{align}} \\ &= (1-\alpha)\, KL(T^s_{\mathcal{U}}||Q_e) &&+\alpha \sum_{f\in F}\sum_{i\in\mathcal{U}^s} ||y_i^e - y_i^f||^2, \end{aligned} \tag{3}$$

where y_i^e is the vertical position of the landmark i in the embedding e that is iteratively optimized. For an embedding $f \in F$ containing landmarks from the same collection $\mathcal{U}^s$, y_i^f is the vertical position of i in f. For inter-collection embeddings, i.e., optimization of $\mathcal{U}^s$ from $\mathcal{V}^s$, y_i^f is computed as the mean position of the landmarks in f that are connected by an edge to the landmark i in e. C_e is the composition of two different costs, the C_e^{tSNE}, as presented in Equation 2, and C_e^{align}, which minimizes the squared distances between the position of a landmarks in the embedding e and in the embedding f. The parameter α controls the weight that is given to the two terms. For $\alpha = 0$ the cost function is the same as a traditional tSNE minimization, while for $\alpha = 1$, only the squared distances along the vertical axes are minimized.

As before, the embeddings are generated by moving the points in the opposite direction of the gradient of C_e. We found that good results are obtained if we optimize all embeddings for $\mathcal{U}$ and $\mathcal{V}$ simultaneously, letting each one influence the other during the minimization. Regarding the parameter α, we found that for our test cases it works well to start with a relatively high value, e.g., $\alpha = 0.5$. In this way, the landmarks are iteratively placed in similar positions along the vertical axes right from the start. However, we believe that the preservation of the similarities between similar vertices as computed by C_e^{tSNE} is of greater importance as it is the main insight that the user aims at achieving **(T1)**. For this reason we linearly reduce the value of α down to 0 after a number of iterations. Empirically, we found that a linear reduction of α to 0 in 500 iterations is a good strategy for all test cases.

5.3. Hierarchy exploration

The data exploration is implemented with a *filter* and *drill-in* strategy that starts from the highest scale in the HSNE hierarchy for both collections $\mathcal{U}$ and $\mathcal{V}$. Landmarks at this level of abstraction represent the main clusters. WAOW-Vis provides a multiscale exploration of the clusters, which is performed by letting the user select a set of landmarks in either of the two collections with a brushing interaction. The selection leads to a refined visualization that contains the influenced landmarks in the lower scale for the corresponding collection.

We now provide the details on the creation of embeddings that contain landmarks in lower scales of the hierarchy starting from a selection in a higher scale for one of the collection, e.g., $\mathcal{U}$. A similar approach is performed if the selection is within $\mathcal{V}$. Given landmarks $\mathcal{U}^s$ at scale s and a set of indices of selected landmarks A, the new visualization contains a subset of landmarks in $\mathcal{U}^{s-1}$, which are under the area of influence of the selection in s. As defined in Section 4.3, the area of influence of the landmarks associated to a scale s is defined by the matrix $I^s(i,j)_{\mathcal{U}}$. $I^s_{\mathcal{U}}$ has size $|\mathcal{U}^{s-1}| \times |\mathcal{U}^s|$, where $I^s(i,j)_{\mathcal{U}}$ is the probability that the landmark $\mathcal{U}^{s-1}_i$ in the previous scale is well represented by $\mathcal{U}^s_j$. The new embedding contains all landmarks $\mathcal{U}^{s-1}_k$ at scale $s-1$ for which the following is true:

$$\sum_{a \in A} I^s(k,a) > \theta, \tag{4}$$

where $0 < \theta \leq 1$ is a user defined threshold. Intuitively, $\sum_{a \in A} I^s(k,a)$ represents the probability for the landmark $\mathcal{U}^{s-1}_k$ to be influence by the selection A of landmarks in $\mathcal{U}^s$. We experimentally found that a default value of $\theta = 0.5$, allows for the effective exploration of the clusters of vertices. Figure 1 shows an example. Once a cluster is selected, its detailed information at the lower scale can be visualized upon the user's request. In the next section, we provide further examples of how the hierarchical exploration of the data give richer insight on the hierarchy of clusters in the graph.

6. Implementation

WAOW-Vis is implemented in C++ for performance reasons and, when possible, heavily uses OpenMP [DM98] to parallelize computations. It fully supports the Progressive Visual Analytics paradigm [FP16, MPG*14], allowing for the visualization of the evolutions of the embeddings, while the embeddings are iteratively generated. Therefore, the user does not have to wait a fixed number of iterations and can autonomously decide on the convergence of the embedding by evaluating their visual stability [PLvdM*16]. The compressed bitmaps are implemented using the C++ version of the Roaring Bitmaps library [LSYKK16]. The modified version of the HSNE algorithm [PHL*16], presented in this work, is derived from the original C++ implementation available in the High-Dimensional-Inspector library [Pez17]. The embeddings are implemented in OpenGL. The word-clouds are implemented in Javascript using D3 and are integrated in the C++ application using the QtWebKit Bridge. Finally, WAOW-Vis is released as part of the High-Dimensional-Inspector library [Pez17].

7. Test cases

To evaluate WAOW-Vis we present real-world examples of the analysis of bipartite graphs of social-network scale. We identified three domains to analyze: computer science, news, and politics. For each one of these domains, we chose as elements of the collection $\mathcal{V}$ a number of Twitter-feeds, i.e., Twitter users that post mainly in the chosen domain. The collection $\mathcal{U}$ contains all the followers for the element in $\mathcal{V}$. Therefore, the resulting bipartite graph encodes the *follower* relationships between the users and the Twitter-streams in the specific domain. In WAOW-Vis, clusters of similar users are communities that share similar interests in the specific domain, while clusters of similar feeds, share similar groups of followers.

We decided to focus on the analysis on the relationship between users and feeds in Twitter as we are interested in exploring the presence of echo-chambers, or filter bubble, in social networks. An echo chamber is a community of users that receives only polarized information concerning a specific domain, e.g., politics. The presence of echo chambers in social networks is deemed responsible for the polarization of the public discourse in recent years [Gar09]. Contrary to other social networks like Facebook, follower relationships on Twitter are openly available through the Twitter-API, allowing us to test WAOW-Vis on real-world data.

Table 1 presents the overview on the datasets that we collected and analyzed. Every column corresponds to a bipartite graph associated with a specific domain. The computer science dataset contains Twitter-feeds associated with several programming languages, e.g., Java, C++, PHP and OpenGL. The news dataset contains journalists and presenters of two of the major United States' news outlets, i.e., CNN and Fox News. The politics datasets contains the Twitter accounts of every United States' senator. The last column presents an additional dataset which is the union of the previously introduced ones. The first three rows present the number of vertices in the collections and the edges connecting them. Note that, for the presented datasets, $|\mathcal{U}| \gg |\mathcal{V}|$ due to scalability issues of the Twitter mining. Twitter imposes a limitation in the number of user-followers relationships, i.e., edges in our graph, that can be obtained per minute. This limit is of 5000 links per minute, hence it required approximately 9 effective days of mining for gathering the datasets.

In Table 1, we present the computation time in seconds for the three processing steps presented in Section 4, together with the maximum memory occupation of WAOW-Vis for each dataset. The results are generated on a workstation with an 3.40GHz Intel i7-2600 CPU and 20 GB of memory. The computation of the hierarchies needed for the analysis of the largest dataset, which contains 19.7M users, takes less than one hour. To give a perspective on size, Twitter has an estimated number of 340M active daily users. We performed a comparison with the traditional HSNE hierarchy computation, which relies on the FLANN library [ML14] for the similarity computation, using the dense representation of the graph. However, the computation of the HSNE hierarchy is impossible to perform for all the datasets due to the heavy memory requirements, hence demonstrating the need for a novel approach as presented in this work.

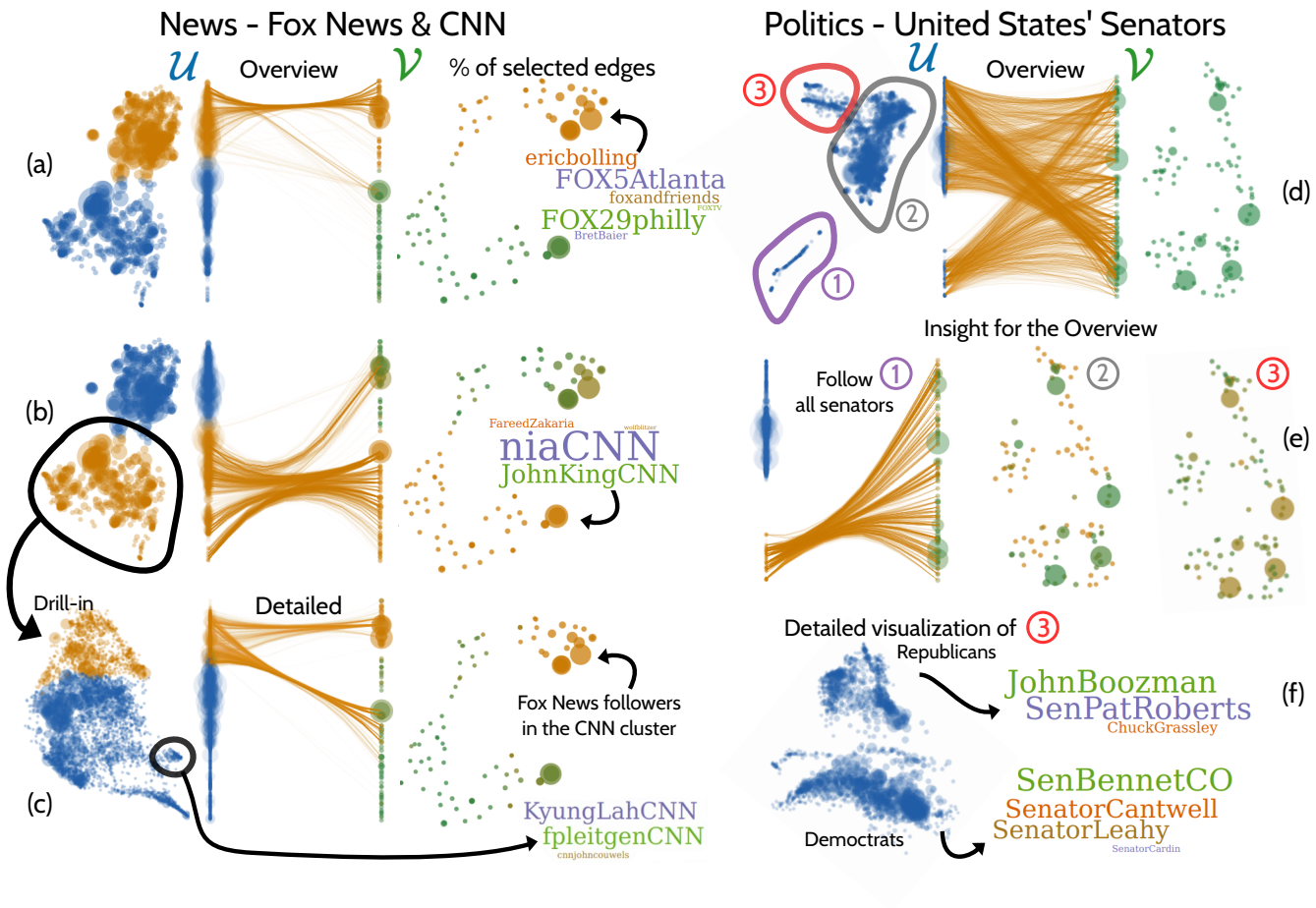

Figure 7: ***Analysis of the United States' politics and news datasets***. *The News dataset comprises a collection $\mathcal{V}$ of Twitter accounts of journalists from two of the United States' major news outlets, CNN and Fox News. Users that follows the feeds in $\mathcal{V}$ are in the collection $\mathcal{U}$. Two echo chambers [Gar09] are identified in the dataset (a,b). Users in the top cluster are following only journalists from Fox News (a), while the ones at the bottom are following only CNN's journalists (b). In the two visualizations (a,b) only the edges linked to the selection are shown. In the right embedding, landmarks are visualized with a green-to-orange color scale that shows the percentage of incoming edges in the current selection. The clusters of orange-colored landmarks in $\mathcal{V}$ confirms the strong association for the selection in $\mathcal{U}$. By drilling into the CNN cluster, a sub community that follows Fox News accounts is identified (c) The analysis of the politics datasets does not show strong evidence of a polarized audience (d). A cluster containing users that follow all the senators is highlighted in purple (d,e). The cluster in gray contains users following the senators with the largest audience, while users in the red cluster follow senators with not so many followers (d,e). A detailed visualization of the red cluster reveal that a polarized audience exists for these senators (f).*

Table 1: ***Datasets*** *information are presented in the columns. The first three rows present the size of the biparite graph, where (M) means millions of elements. Computation time in seconds for the SIT creation, kNN and HSNE computation are presented. Finally, peak memory occupation of WAOW-Vis is given.*

	Computer S.	News	Politics	Combined
$\lvert\mathcal{U}\rvert$	1.97M	7.67M	12.42M	19.7M
$\lvert\mathcal{V}\rvert$	145	83	100	329
#Edges	4.4M	10.8M	24.5M	38.9M
SIT (s)	13	34	63	283
*k*NN (s)	50	59	430	1960
HSNE (s)	13	11	60	202
Mem. (GB)	1.6	2.1	3.7	10.6

7.1. News dataset

Figure 7a-b shows the exploratory analysis of the news dataset performed using WAOW-Vis. Two separated clusters of Twitter feeds are visible in the right embedding, i.e., $\mathcal{V}$. By visualizing the owners of the Twitter feeds in the word cloud, we realize that they belong to Fox News journalists for the cluster at the top and to CNN journalists for the one at the bottom. This insight is confirmed by the visualization of the edges connecting the two collections. Contrary to the visualizations that we presented so far, in Figure 7a-b we show the edges related to a user-defined selection of landmarks. Selected landmarks in the left embedding are rendered with a shade of orange, and only edges connected to these landmarks are shown. Most of these edges are connected to the top cluster in the embedding on the right. Here, landmark colors encode the percentage

of incoming edges that are currently selected by the user with a green-to-orange color scale. The top cluster in the right embedding of Figure 7a has the same shade of orange of the selection in the left embedding. This means that the current user selection among $\mathcal{U}$ is mainly connected to the top cluster in $\mathcal{V}$. The same observation can be done for the cluster at the bottom of the visualization, as shown in Figure 7b.

For both selections, only a small number of edges are connected to the opposite cluster. This insight leads to the conclusion that two echo chambers [Gar09] exist for the two news outlets. However, in Figure 7b, we can observe that a more consistent stream of edges is connecting CNN followers to the Fox News feeds. In order to reveal more sub-communities within the cluster, a detailed visualization is generated by drilling into the hierarchy. Figure 7c shows the resulting embedding. The selection in the embedding contains all the landmarks (i.e., group of users) that follow Fox News' accounts. Finally, by selecting the small cluster which is encircled in Figure 7, followers of international CNN reporters, such as Kyung Lah and Frederik Pleitgen, are identified.

7.2. Politics dataset

Figure 7d-f shows the exploratory analysis of the politics dataset. In this test case, we expected to see an echo chamber for users connected to the Republican senators and one for those connected to the Democratic senators. However, a clear separation for the collection $\mathcal{V}$ is not visible in the visualization shown in Figure 7d. To better understand the interrelationships between the two collections, the user interacts with WAOW-Vis in order to get a more detailed insight on the visible clusters. The cluster ① in Figure 7d is the most distinct one. To understand the connections of the clusters ① to the collection $\mathcal{V}$, the edges linked to it are visualized. Figure 7e shows that the selected landmarks are connected to the accounts of every senators. We conclude that this clusters contain political enthusiasts or, more likely, software-controlled Twitter accounts. These accounts, also known as Twitter-bots, work as tweet aggregators for a specific domain, for example, by automatically reposting the senators' tweets.

The cluster ② is then selected by the user. A visualization of the percentage of the incoming edges in the right embedding is visualized in Figure 7e. Only the large points in the embedding are colored with a shade of orange. These points correspond to the senators with the largest user base, such a senator Elisabeth Warren and Marco Rubio. This result shows that the cluster ② identifies the community of users who are following mainly the most famous politicians. Finally, the cluster ③ in Figure 7d corresponds to users following senators with a much smaller user base, as can be seen by the result of the selection in Figure 7e. In the overview, cluster ③ already shows a separation in two sub clusters. A detailed visualization, which is generated by drilling in the hierarchical representation, shows a better separation of these clusters. The resulting embedding, which is presented in Figure 7f, shows that a polarization of the users exists for Republicans and Democrats in this sub community.

8. Conclusions and Future Work

In this work we have presented Who's-Active-On-What-Visualization (WAOW-Vis), a visual analytics system for the exploratory analysis of large bipartite graphs. We presented a novel graph preprocessing pipeline that is inspired by the recent developments in the analysis of large high-dimensional data. The scalability of WAOW-Vis is enabled by three main contributions of this work. The adoption of compressed bitmaps for representing the graph and the novel Sets Intersection Tree (SIT) for efficiently computing Jaccard similarities between the bitmaps. The similarities are then used to generate a hierarchical representation of the graph with a modified version of the Hierarchical Stochastic Neighbor Embedding (HSNE). Moreover, we presented several insights obtained by the exploratory analysis of large datasets that we mined from Twitter.

WAOW-Vis, however, does not come without limitations. First, while WAOW-Vis can handle very large bipartite graphs, it can only handle undirected and unweighted graphs. Extending our technique to handle weighted and directed graphs is an interesting future work. Furthermore, in the exploratory analysis of the data, the several visualizations that are generated make it difficult to keep a mental mapping of the exploration process. Höllt et al. recently proposed CyteGuide [HPvU*17] for guiding the user in the exploration of a single HSNE hierarchy. An interesting future work is the development of a similar approach for guiding the data exploration in WAOW-Vis. Moreover, the visualizations of the links between the two collections is limited to the 1-dimensional embeddings. This may be a limitation as most of the insights are obtained through the analysis of the 2-dimensional embeddings. Finally, in the test cases we analyzed datasets where the two collections are very different in size, due to the query limit imposed by Twitter. Because we are not limited to this kind of datasets, we are interested in experimenting with more balanced bipartite graphs. In the immediate future, we plan to introduce the visualization of the links between the 2d embeddings as presented by Collins and Carpendale [CC07]. Finally, bipartite graphs are widely used in biomedical research [HSBW11] and for the visualization of deep neural networks [MCZ*17]. We are interested in exploring the insights that WAOW-Vis can provide to these fields.

References

[BBD*11] Batagelj V., Brandenburg F. J., Didimo W., Liotta G., Palladino P., Patrignani M.: Visual analysis of large graphs using (x, y)-clustering and hybrid visualizations. *IEEE transactions on visualization and computer graphics 17*, 11 (2011), 1587–1598. 3

[BBHR*16] Behrisch M., Bach B., Henry Riche N., Schreck T., Fekete J.-D.: Matrix Reordering Methods for Table and Network Visualization. *Computer Graphics Forum 35* (2016), 24. 2, 3

[BDF*10] Bezerianos A., Dragicevic P., Fekete J.-D., Bae J., Watson B.: GeneaQuilts: A System for Exploring Large Genealogies. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Oct. 2010), 1073–1081. URL: https://hal.inria.fr/inria-00532939, doi:10.1109/TVCG.2010.159. 3

[BGW03] Brandes U., Gaertler M., Wagner D.: *Experiments on graph clustering algorithms*. Springer, 2003. 3

[CC07] Collins C., Carpendale S.: Vislink: Revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics 13* (2007), 1192–1199. 3, 10

[DM98] DAGUM L., MENON R.: Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE 5*, 1 (1998), 46–55. 8

[DRM12] DUMAS M., ROBERT J.-M., MCGUFFIN M. J.: Alertwheel: radial bipartite graph visualization applied to intrusion detection system alerts. *IEEE Network 26*, 6 (2012). 3

[DRRD12] DÖRK M., RICHE N. H., RAMOS G., DUMAIS S.: Pivotpaths: Strolling through faceted information spaces. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2709–2718. 3

[EDG*08] ELMQVIST N., DO T.-N., GOODELL H., HENRY N., FEKETE J.-D.: Zame: Interactive large-scale graph visualization. In *Pacific Visualization Symposium (PacificVis)* (2008), IEEE, pp. 215–222. 3

[FP16] FEKETE J.-D., PRIMET R.: Progressive analytics: A computation paradigm for exploratory data analysis. *arXiv preprint arXiv:1607.05162* (2016). 8

[Gar09] GARRETT R. K.: Echo chambers online?: Politically motivated selective exposure among internet news users. *Journal of Computer-Mediated Communication 14* (2009), 265–285. 8, 9, 10

[Gey11] GEYER C.: Introduction to markov chain monte carlo. *Handbook of Markov Chain Monte Carlo* (2011), 3–48. 6

[GFC05] GHONIEM M., FEKETE J.-D., CASTAGLIOLA P.: On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization 4*, 2 (2005), 114–135. 2, 3, 6

[GKL*13] GHANI S., KWON B. C., LEE S., YI J. S., ELMQVIST N.: Visual analytics for multimodal social network analysis: A design study with social scientists. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2032–2041. 3

[HF06] HENRY N., FEKETE J.-D.: Matrixexplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 677–684. 3

[HFM07] HENRY N., FEKETE J.-D., MCGUFFIN M. J.: Nodetrix: a hybrid visualization of social networks. *IEEE transactions on visualization and computer graphics 13*, 6 (2007), 1302–1309. 3

[HMM00] HERMAN I., MELANÇON G., MARSHALL M. S.: Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics 6*, 1 (2000), 24–43. 1, 2

[Hol06] HOLTEN D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 741–748. 3

[HPvU*17] HÖLLT T., PEZZOTTI N., VAN UNEN V., KONING F., LELIEVELDT B. P., VILANOVA A.: Cyteguide: Visual guidance for hierarchical single-cell analysis. *IEEE Transactions on Visualization and Computer Graphics 24* (2017). 10

[HSBW11] HEINRICH J., SEIFERT R., BURCH M., WEISKOPF D.: Bicluster viewer: a visualization tool for analyzing gene expression data. *Advances in Visual Computing* (2011), 641–652. 2, 3, 10

[HSS15] HADLAK S., SCHUMANN H., SCHULZ H.-J.: A survey of multi-faceted graph visualization. In *Eurographics Conference on Visualization (EuroVis).* (2015), pp. 1–20. 3

[HVW09] HOLTEN D., VAN WIJK J. J.: Force-directed edge bundling for graph visualization. In *Computer Graphics Forum* (2009), vol. 28, pp. 983–990. 6

[JHGH08] JIA Y., HOBEROCK J., GARLAND M., HART J.: On the visualization of social and other scale-free networks. *IEEE transactions on visualization and computer graphics 14*, 6 (2008), 1285–1292. 3

[LF06] LESKOVEC J., FALOUTSOS C.: Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (2006), pp. 631–636. 3

[LPS*06] LEE B., PLAISANT C., SIMS C., FEKETE J.-D., HENRY N.: Task taxonomy for graph visualization. In *BELIV '06: Proceedings of the 2006 AVI workshop on BEyond time and errors* (Venezia, Italy, May 2006), ACM, ACM, pp. 1–5. URL: https://hal.inria.fr/hal-00851754, doi:10.1145/1168149.1168168. 4

[LSYKK16] LEMIRE D., SSI-YAN-KAI G., KASER O.: Consistently faster and smaller compressed bitmaps with roaring. *Software: Practice and Experience 46*, 11 (2016), 1547–1569. 2, 5, 8

[MAH*12] MARTINS R. M., ANDERY G. F., HEBERLE H., PAULOVICH F. V., DE ANDRADE LOPES A., PEDRINI H., MINGHIM R.: Multidimensional projections for visual analysis of social networks. *Journal of Computer Science and Technology 27* (2012), 791–810. 3

[MCZ*17] MING Y., CAO S., ZHANG R., LI Z., CHEN Y., SONG Y., QU H.: Understanding hidden memories of recurrent neural networks. *arXiv preprint arXiv:1710.10777* (2017). 3, 10

[MHG10] MCCANDLESS M., HATCHER E., GOSPODNETIC O.: *Lucene in Action: Covers Apache Lucene 3.0*. Manning Publications Co., 2010. 5

[ML14] MUJA M., LOWE D.: Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence 36*, 11 (2014), 2227–2240. 6, 8

[MO04] MADEIRA S. C., OLIVEIRA A. L.: Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 1*, 1 (2004), 24–45. 2, 3

[MPG*14] MÜHLBACHER T., PIRINGER H., GRATZL S., SEDLMAIR M., STREIT M.: Opening the black box: Strategies for increased user involvement in existing algorithm implementations. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1643–1652. 8

[OKHC14] OGHABIAN A., KILPINEN S., HAUTANIEMI S., CZEIZLER E.: Biclustering methods: biological relevance and application in gene expression analysis. *PloS one 9*, 3 (2014), e90801. 2, 3

[Pez17] PEZZOTTI N.: High dimensional inspector, 2017. URL: https://github.com/Nicola17/High-Dimensional-Inspector. 6, 8

[PGAR15] PONTES B., GIRÁLDEZ R., AGUILAR-RUIZ J. S.: Biclustering on expression data: A review. *Journal of biomedical informatics 57* (2015), 163–180. 2, 3

[PHL*16] PEZZOTTI N., HÖLLT T., LELIEVELDT B., EISEMANN E., VILANOVA A.: Hierarchical stochastic neighbor embedding. In *Computer Graphics Forum* (2016), vol. 35, pp. 21–30. 2, 4, 6, 8

[PLvdM*16] PEZZOTTI N., LELIEVELDT B., VAN DER MAATEN L., HOLLT T., EISEMANN E., VILANOVA A.: Approximated and user steerable tsne for progressive visual analytics. *IEEE Transactions on Visualization and Computer Graphics PP*, 99 (2016), 1–1. 8

[RC67] ROBINSON A. H., CHERRY C.: Results of a prototype television bandwidth compression scheme. *Proceedings of the IEEE 55*, 3 (1967), 356–364. 4, 5

[SGL08] STASKO J., GÖRG C., LIU Z.: Jigsaw: supporting investigative analysis through interactive visualization. *Information visualization 7*, 2 (2008), 118–132. 3

[Tam07] TAMASSIA R.: *Handbook of Graph Drawing and Visualization (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2007. 3

[TLZM16] TANG J., LIU J., ZHANG M., MEI Q.: Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web* (2016), pp. 287–297. 4

[vdMH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-SNE. *Journal of Machine Learning Research 9*, 2579-2605 (2008), 85. 2, 7

[VLKS*11] VON LANDESBERGER T., KUIJPER A., SCHRECK T., KOHLHAMMER J., VAN WIJK J. J., FEKETE J.-D., FELLNER D. W.: Visual analysis of large graphs: state-of-the-art and future research challenges. In *Computer Graphics Forum* (2011), vol. 30, pp. 1719–1749. 1, 2, 3

[XCQS16] XU P., CAO N., QU H., STASKO J.: Interactive visual co-cluster analysis of bipartite graphs. In *Pacific Visualization Symposium (PacificVis)* (2016), IEEE, pp. 32–39. 3

[ZXW*16] ZAHARIA M., XIN R. S., WENDELL P., DAS T., ARMBRUST M., DAVE A., MENG X., ROSEN J., VENKATARAMAN S., FRANKLIN M. J., ET AL.: Apache spark: A unified engine for big data processing. *Communications of the ACM 59*, 11 (2016), 56–65. 5

[ZXYQ13] ZHOU H., XU P., YUAN X., QU H.: Edge bundling in information visualization. *Tsinghua Science and Technology 18*, 2 (2013), 145–156. 3