EG UK Computer Graphics & Visual Computing (2023)
D. Hunter and P. Vangorp (Editors)

SunburstChartAnalyzer: Hierarchical Data Retrieval from Images of Sunburst Charts for Tree Visualization

Prakhar Rastogi, Karanveer Singh and Jaya Sreevalsan-Nair †

Graphics-Visualization-Computing Lab,
International Institute of Information Technology Bangalore, India

Abstract

Data extraction from visualization is a challenging problem in computer vision owing to the huge "design space of possible vis idioms." Different visualizations pose different challenges in automated data extraction from their images, which is needed in document analysis. In the case of sunburst charts for hierarchical data, the extracted data has to be also correctly organized as a tree data structure. Overall, data extraction has to consider different components of a chart image, such as text, annular sectors, levels, etc., and their ordering. We propose an end-to-end algorithm, SunburstChartAnalyzer, for data extraction from sunburst charts. The algorithm includes chart classification, component extraction, and hierarchical data organization. We further propose a composite metric to evaluate the correctness of SunburstChartAnalyzer. Our experimental results show that our proposed method works for trees of all sizes, and particularly well for shallow and medium-depth trees.

Keywords: *Hierarchical data, Visualization, Sunburst charts, Circle objects, Tree data structure, Optical Character Recognition, Text detection, Hough transform, Geometry*

CCS Concepts

• ***Human-centered computing*** → ***Visualization techniques;*** *Accessibility systems and tools;* • ***Computing methodologies*** → ***Information extraction; Image processing;***

1. Introduction

The proliferation in the use of charts in various data science applications has led to active research in automated data retrieval from these charts [DSN22, DDSN21a, DDSN21b, CJP*19]. The data retrieval requires a combination of image processing and computer vision methods where it is chart-specific, owing especially to the huge "design space of possible vis idioms" [Mun14]. Sunburst is a widely used hierarchical data visualization that uses concentric circles to encode hierarchical levels, grows outwards with the tree, and its annular sectors represent tree nodes. With their complex geometric structure, these chart images pose a challenge to extract the circular object geometry to retrieve the data and organize it in the tree format. At the same time, its structured geometry can be exploited for the task of automated data extraction.

Here, we propose SunburstChartAnalyzer, an end-to-end algorithm that involves image processing and machine learning methods for efficient and accurate data retrieval from sunburst chart images. For validation, we propose the use of *tree edit distance* to compare the source and extracted data of sunburst images that consider both labels (text) and tree structure. We run our experiments on sunburst chart images taken from the internet as well as synthetically generated for qualitative and quantitative analysis, respectively. Our proposed workflow and sample results are shown in Figures 1 and 2, respectively.

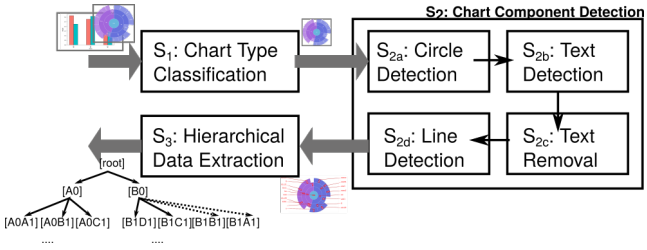

Figure 1: *Our proposed workflow for SunburstChartAnalyzer for hierarchical data retrieval from sunburst chart images.*

One of the motivations behind this line of work is to improve the accessibility of charts for the visually challenged, required in document analysis [CJP*19]. Similar work has been done for scientific diagrams [RAK*20], and its extension to sunburst images is novel.

† Thanks to Machine Intelligence and Robotics (MINRO) grant from the Government of Karnataka and IIITB for funding. email: jnair@iiitb.ac.in

DOI: 10.2312/cgvc.20231200

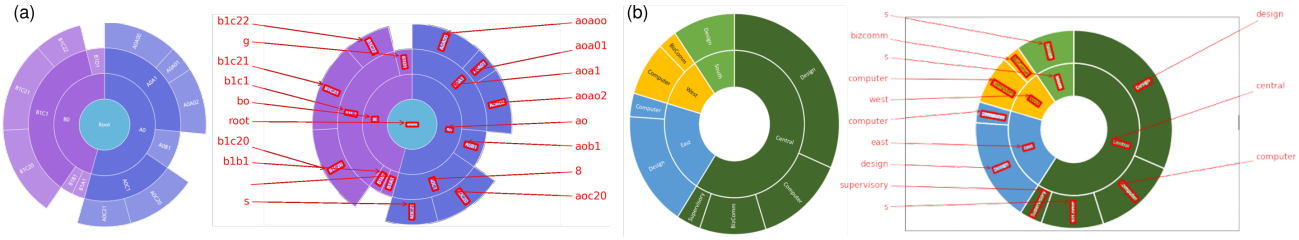

Figure 2: *Results from SunburstChartAnalyzer with source image of sunburst and its annotated version for sunburst charts (a) with and (b) without a filled center ((b) has an unlabeled root), respectively. (Input image source: from the Internet).*

2. Methodology

SunburstChartAnalyzer has three key steps, namely, ($\mathbb{S}_1$) chart type classification, ($\mathbb{S}_2$) chart component detection, and ($\mathbb{S}_3$) hierarchical data extraction.

$\mathbb{S}_1$ – Chart Type Classification: We first determine the chart type from the input image, such that, if it is of sunburst type, it proceeds with the workflow, otherwise it is rejected. We use supervised machine learning and deep learning models in our chart type classifier. We compare the performance of different models to determine the optimal one for our workflow.

Given the paucity of sunburst chart images, we first curate the training dataset using images generated from synthetic data. We ensure the class balance in the dataset, and also specifically include other charts with circular objects, *e.g.,* pie charts, scatter plots, and bubble plots, in the dataset to improve the classification of sunburst.

$\mathbb{S}_2$ – Chart Component Detection: Chart component detection has four key sequential steps, namely, ($\mathbb{S}_{2a}$) circle detection, ($\mathbb{S}_{2b}$) text detection, ($\mathbb{S}_{2c}$) text removal, and ($\mathbb{S}_{2d}$) line detection.

($\mathbb{S}_{2a}$) Circle Detection: The most observable feature of a sunburst chart is the concentric circles with its *truncated* or *annular* sectors. While Circle Hough Transform (CHT) [DH72] is suitable for extracting a single circle, it does not perform as desired for concentric circles. It is difficult to generalize parameters for CHT across different circles. Hence, we use CHT to extract one circle to determine its center. We then use Line Hough Transform (LHT) [DH72] to find partitions or separators between sectors.

($\mathbb{S}_{2b}$) Text Detection: Text must be detected and removed prior to data extraction from the geometry in the image. We use the oft-used optical character recognition (OCR) for text detection. The output of this step is textboxes and its positions, which are essential to determine the hierarchy encoded in the sunburst. For every detected textbox, we calculate its distance from the center of the circle obtained in $\mathbb{S}_{2a}$. We sort these distances and group the proximate textboxes to find *siblings* in the tree data structure. This is the first cut computation of the hierarchical levels in the tree data.

($\mathbb{S}_{2c}$) Text Removal: Here, care is taken to remove text alone and retain its background in the sectors. This ensures no artificial discontinuities in the geometry that could otherwise potentially interfere with its extraction. Thus, the desired outcome is the source image without any text rendering. We use text inpainting [BBS01, KB12] to restore the background pixels (Figure 3).

We first determine the size of the textboxes detected in $\mathbb{S}_{2b}$ to create white rectangular masks for textboxes. The remaining image is rendered black, thus creating a binary image (Figure 3(a)). Our approach is to create a correctly oriented and symmetric mask about the textbox-center. We initially use the length of the textbox through its center as the length of a line mask, and the line is padded by the width of the textbox, again taken at its center (Figure 3(left)). However, we observe that such a mask interferes with the borders of the sunburst chart, thus affecting our line detection in $\mathbb{S}_{2d}$.

Hence, we improve the mask by using the corners of the textbox. We calculate the distances between the corners/vertices of the rectangle and the mid-points of the lines joining them. We then use the ratio of the largest and the shortest distance to determine the endpoints of the line mask and its padding to cover the textbox entirely to obtain compact masks (Figure 3 (right)). Thus, our optimization step provides a mask correction by visibly maximizing the length of masking lines and minimizing the thickness for each text detected. We observe that this modified mask improves the detection of vertically oriented text especially.

($\mathbb{S}_{2d}$) Line Detection: Given the limitation of CHT in extracting concentric circles, it is important to accurately detect lines that function as sector separators. SunburstChartAnalyzer is based on the assumption that these lines can be detected more accurately and contiguously in the text-free image. Line detection entails standard image pre-processing steps on the text-free image. We convert the image to grayscale, apply slight blurring to it, and finally use Canny edge detection to get the edge image (Figure 4(a)).

We now apply the probabilistic LHT (PHT) to the edge image [KEB91, MGK00] to localize and assemble the various line segments/edges in the source image. We observe that PHT also gives noise that is filtered out using a specific criterion. This criterion is that we retain the *good* lines if the pixels satisfy a line equation. We also check if the line segment is not farther than 20 pixels from the circle center from $\mathbb{S}_{2a}$, as a heuristic thresholding step. This step gives desired results as shown in Figure 4(b)-(c).

Using these filtered lines, the steps for extracting hierarchical levels (Figure 4(d)) are:

1. For each hierarchical level, we determine the equation of the (concentric) circle passing through the midpoint of the diagonal of the textboxes detected in $\mathbb{S}_{2b}$.

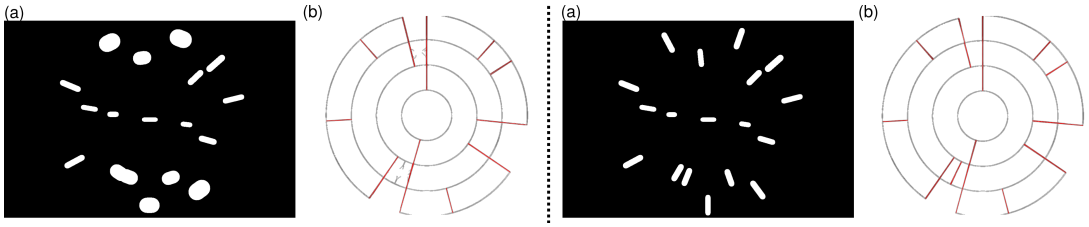

Figure 3: *Text removal from the image in (a) to give the output in (b), using (left) initial mask and (right) optimized mask.*

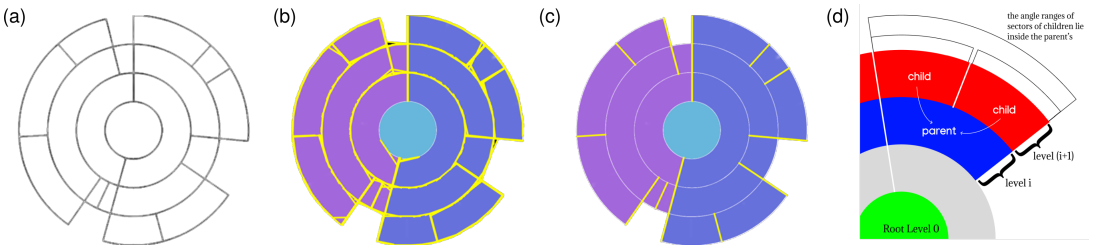

Figure 4: *Line detection using (a) Canny edge detection, and (b,c) our line detection algorithm before and after filtering out the noise. Extraction of hierarchical levels in* $\mathbb{S}_{2d}$ *is diagrammatically explained in (d).*

2. Using the obtained circle equation for each concentric circle/level and the inside-outside test on the endpoints of the line segments, we find all the line segments that intersect the circle.
3. We store for each hierarchical level its corresponding intersecting lines. The levels start from the root as 0 and are incremented by 1 for every circle that occurs in the direction of a radial line moving away from the center.

We also remove duplicate lines which are detected by checking the angle between any two extracted lines. We compute these pairwise angles using trigonometric relationships and heuristically filter out lines that make the pairwise angle less than 3.5^{o}, thus eliminating the longer lines exclusively.

$\mathbb{S}_3$ – Hierarchical Data Extraction: Using the text and hierarchical levels, we now assign parents to each of the extracted texts. These textboxes are equivalent to the *nodes* of the tree data. Every node represents an annular sector and has the following attributes: (i) text, (ii) the angle subtended by the text at the circle center, (iii) the parent and children of that node, (iv) the angle range spanned by the node and (v) the ratio of the sector area to that of the circle.

In a few of the sunburst chart images, we encounter a different but specific rendering style of an unfilled and unlabeled innermost circle for the root node (Figure 2(b)). In such charts, we first fill the innermost circle, thus, introducing a dummy root node with empty text, prior to applying $\mathbb{S}_1$-$\mathbb{S}_3$.

We thus construct the tree from the inside outwards, and assign a parent to each node recursively. In sunburst, the sector angle of the parent is fragmented by those of the children (Figure 4(d)). Hence, we find a node with a larger angle range within which the angle range of a concerned node lies, thus establishing that the former is the parent of the latter. A detailed explanation of how the tree construction algorithm organizes data is as follows:

1. For every node in level i, we find the pair of consecutive sector separators in level $(i+1)$ that physically contain the node. This is done by comparing the angle ranges assigned to the pair of nodes in levels i and $(i+1)$.
2. Whenever such a pair is encountered, we assign the node in level i as the parent of the node in level $(i+1)$.
3. Repeating #1 and #2 for all levels moving outwards from the root, we obtain the tree representation of the data extracted from the sunburst chart image.

Evaluation and Validation: For evaluation and validation of our proposed algorithm, we use the tree edit distance (TED) [ZSS92] and its corresponding similarity score [SGAM*15] as a metric. TED is the cost of converting one tree into another through the permissible operations: (i) insertion of a node, (ii) deletion of a node and (iii) renaming of a node. Each of these operations has a cost assigned to it using which the final cost (the tree edit distance) is calculated. We have used the TED values to compute the similarity between the original tree and the tree extracted from our algorithm. The similarity score based on TED [ZSS92, SGAM*15] between trees T_i and T_j, with $d = \text{TED}(T_i, T_j)$ is:

$$\text{sim}(T_i, T_j) = \frac{1}{\sqrt{1+d}}.$$

Here, we calculate two different kinds of similarities based on TED: **Structural Similarity (SS)**, and **Data Similarity (DS)**. The SS is a measure of how similar the structure of the extracted tree and the original tree is. We calculate SS by only taking into account the costs of inserting and deleting a node. The DS is a measure of how accurately is the data being extracted. It is calculated using TED based only on the cost of renaming a node between two trees.

Implementation: For $\mathbb{S}_1$, we created the training dataset for chart classification. It contains 400 images comprising five different types of charts, of which 200 images are of sunburst charts, 50

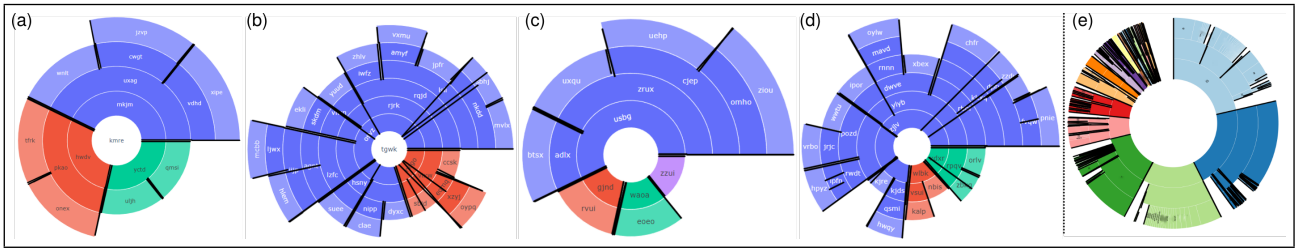

Figure 5: *Results of SunburstChartAnalyzer for line detection in a sample of the images of a sunburst (a,b) with filled and (c,d) unfilled centers/unlabeled roots, and created with (a,c) shallow (15 nodes) and (b,d) deep (35 nodes) trees (Input image source: synthetically generated by authors). (e) Our algorithm fails for a highly fragmented tree (Input image source: [YCHL22]).*

of bar charts, 50 of pie charts, 50 of scatter plots, and 50 of bubble plots. The images are equally downloaded from the internet and synthetically generated using input datasets and third-party visualization libraries (*e.g.,* Plotly [Plo15]).

For $\mathbb{S}_{2b}$, we specifically used Keras OCR [C*15]. For $\mathbb{S}_{2c}$ and $\mathbb{S}_{2d}$, we used the functions `cv2.inpaint` and `cv2.HoughLinesP` from OpenCV library [Bra00]. For evaluation and validation, we used the Zhang-Shasha tree edit distance [ZZL*14] in the zss Python module [zss]. Before calculating the tree edit distance for the original and extracted tree, we first sorted each level of both trees alphabetically. This was done because the Zhang-Shasha tree edit distance [ZZL*14] algorithm treats trees with same structure but a different order of children as different. Thus sorting would lead to consistency in the order of children in both trees. Thus resulting in accurate edit distances. .The source code of this paper is at the GitHub Repository `http://bit.ly/SunburstChartAnalyzer`.

3. Experiments & Results

We experimented on Convolutional Neural Networks (CNN) and Support Vector Machines (SVM) for $\mathbb{S}_1$. We designed a 6-layer CNN, with 4 convolutional layers and 2 fully connected layers. The test/train ratio split of the dataset also ensures class balance across the five chart types. We got accuracies in the range of 50-97% for CNN depending on the weight initialization of the model, for which we attribute the inconsistency in accuracy to the small size of the training dataset. Thus, we created an SVM model with a parameter setting of `C=1`, `kernel=linear`, and `gamma=auto`. SVM works only on a single-dimensional image. Hence, we converted the RGB color images to grayscale. On using stratified 5-fold Cross Validation for evaluating our model, we got an average accuracy of 86.5% and an F1 score of 0.869, thus proving the SVM is an appropriate classification model for our workflow. We implemented the SVM and CNN using the scikit-learn library [P*11] and the TensorFlow library [A*15], respectively.

Our results for a sample image sourced from the internet and from synthetic images are in Figures 2 and 5, respectively. For the quantitative results of similarity scores, we synthetically generated images of sunbursts from data presented as tree data structures. We grouped the data/charts into two types of categories: (i) filled vs unfilled centers for the root node, and (ii) different tree depths, *i.e.,* deep, medium, and shallow for trees with 35, 25, and 15 nodes, respectively. The validation datasets are generated with an automated script using the Plotly library [Plo15]. For our validation, we used 10 images in each group. Table 1 shows the group averages of similarity scores (SS).

Table 1: *Average Structural Similarity (SS) obtained for each category, calculated from the output of SunburstChartAnalyzer and the input data used for synthetically generated images. For each tree size and center type, 10 images were used for evaluation.*

Image Group	Deep Tree 35 nodes	Medium Tree 25 nodes	Shallow Tree 15 nodes
Filled Center	0.51	0.71	0.75
Unfilled Center	0.64	0.81	0.80

We observed that data similarity (DS) scores are heavily dependent on the quality of the chart images, which influences the quality of the OCR text detection results in $\mathbb{S}_{2b}$. DS score is consistently 1.0 for high-resolution images, *e.g.,* those in Table 1, but is inconsistent for low-resolution (LR) images sourced from the Internet. The choice of OCR software also influences the DS scores; *e.g.,* keras OCR does not perform well for LR inputs.

SunburstChartAnalyzer works well for shallow and medium-depth trees. But the similarity scores to the input data (SS) deteriorate for deeper trees, where the sector separators get crowded and the accuracy of line detection reduces. We show this in an extreme case of an over-fragmented tree (Figure 5(e)).

4. Conclusions

SunburstChartAnalyzer is generalized for various types of sunburst charts and extracts data from them in tree format. Our contributions are in the classification of chart images using learning models, data extraction using image processing and geometric methods, and automated plot generation for creating image datasets. Our method works in all cases except those with high fragmentation, which will be addressed in the future. Also, we can specifically improve the chart type classification using CNNs, and text detection using state-of-the-art techniques (*e.g.,* Google Cloud Vision API [MCF*16]). The extracted text can be used for generating text summaries, it can be used as alt text in online documents, or it can be used with text-to-speech converters to improve the accessibility of the graph for visually impaired people.

References

[A*15] ABADI M., ET AL.: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from https://www.tensorflow.org/. URL: https://www.tensorflow.org/. 4

[BBS01] BERTALMIO M., BERTOZZI A. L., SAPIRO G.: Navier-stokes, fluid dynamics, and image and video inpainting. In *Proceedings of the 2001 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)* (2001), vol. 1, IEEE, pp. I–I. 2

[Bra00] BRADSKI G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000). 4

[C*15] CHOLLET F., ET AL.: Keras, 2015. https://github.com/fchollet/keras – Last accessed on February 15, 2023. URL: https://github.com/fchollet/keras. 4

[CJP*19] CHOI J., JUNG S., PARK D. G., CHOO J., ELMQVIST N.: Visualizing for the Non-Visual: Enabling the Visually Impaired to Use Visualization. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 249–260. doi:10.1111/cgf.13686. 1

[DDSN21a] DADHICH K., DAGGUBATI S. C., SREEVALSAN-NAIR J.: BarChartAnalyzer: Digitizing Images of Bar Charts. In *Proceedings of 1st International Conference on Image Processing and Vision Engineering (IMPROVE)* (2021), INSTICC, SciTePress, pp. 17–28. doi:10.5220/0010408300170028. 1

[DDSN21b] DADHICH K., DAGGUBATI S. C., SREEVALSAN-NAIR J.: ScatterPlotAnalyzer: Digitizing Images of Charts Using Tensor-based Computational Model. In *International Conference on Computational Science, Computational Science – ICCS 2021, Part V, Lecture Notes in Computer Science, volume 12746* (Cham, 2021), Paszynski M., Kranzlmüller D., Krzhizhanovskaya V. V., Dongarra Jack J.and Sloot P. M., (Eds.), Springer International Publishing, pp. 70–83. doi:10.1007/978-3-030-77977-1_6. 1

[DH72] DUDA R. O., HART P. E.: Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM (CACM) 15*, 1 (1972), 11–15. 2

[DSN22] DAGGUBATI S. C., SREEVALSAN-NAIR J.: ACCirO: A System for Analyzing and Digitizing Images of Charts with Circular Objects. In *Computational Science – ICCS 2022, Proceedings of the 22nd International Conference, Part III, chapter 50* (Cham, 2022), Springer International Publishing, pp. 605–612. doi:10.1007/978-3-031-08757-8_50. 1

[KB12] KHODADADI M., BEHRAD A.: Text localization, extraction and inpainting in color images. In *20th Iranian Conference on Electrical Engineering (ICEE2012)* (2012), IEEE, pp. 1035–1040. 2

[KEB91] KIRYATI N., ELDAR Y., BRUCKSTEIN A. M.: A probabilistic Hough transform. *Pattern Recognition 24*, 4 (1991), 303–316. 2

[MCF*16] MULFARI D., CELESTI A., FAZIO M., VILLARI M., PULIAFITO A.: Using Google Cloud Vision in assistive technology scenarios. In *2016 IEEE symposium on computers and communication (ISCC)* (2016), IEEE, pp. 214–219. 4

[MGK00] MATAS J., GALAMBOS C., KITTLER J.: Robust detection of lines using the progressive probabilistic Hough transform. *Computer Vision and Image Understanding 78*, 1 (2000), 119–137. 2

[Mun14] MUNZNER T.: *Visualization Analysis and Design*. CRC press, 2014. 1

[P*11] PEDREGOSA F., ET AL.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830. 4

[Plo15] PLOTLY TECHNOLOGIES INC.: Collaborative Data Science, 2015. Last accessed on February 15, 2023. URL: https://plot.ly. 4

[RAK*20] ROY A., AKROTIRIANAKIS I., KANNAN A. V., FRADKIN D., CANEDO A., KONERIPALLI K., KULAHCIOGLU T.: Diag2graph: representing deep learning diagrams in research papers as knowledge graphs. In *2020 IEEE International Conf. on Image Processing (ICIP)* (2020), IEEE, pp. 2581–2585. 1

[SGAM*15] SIDOROV G., GÓMEZ-ADORNO H., MARKOV I., PINTO D., LOYA N.: Computing text similarity using tree edit distance. In *2015 Annual Conf. of the North American Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conf. on Soft Computing (WConSC)* (2015), IEEE, pp. 1–4. 3

[YCHL22] YE S., CHEN D., HAN S., LIAO J.: 3D Question Answering. *IEEE Transactions on Visualization and Computer Graphics (early access)* (2022), 1–16. doi:10.1109/TVCG.2022.3225327. 4

[zss] zss 1.1.4. https://pypi.org/project/zss/1.1.4/ – Released: May 26, 2016. URL: https://pypi.org/project/zss/1.1.4/. 4

[ZSS92] ZHANG K., STATMAN R., SHASHA D.: On the editing distance between unordered labeled trees. *Information Processing Letters 42*, 3 (1992), 133–139. 3

[ZZL*14] ZHENG W., ZOU L., LIAN X., WANG D., ZHAO D.: Efficient graph similarity search over large graph databases. *IEEE Transactions on Knowledge and Data Engineering (TKDE) 27*, 4 (2014), 964–978. 4