# Tutorial on Graph Stream Analytics

András Benczúr, Ferenc Béres, Domokos Kelen*
{benczur,beres,kdomokos}@sztaki.hu
Institute for Computer Science and Control, Hungary

Róbert Pálovics
palovics@stanford.edu
Stanford University, California, USA

## ABSTRACT

In this short tutorial, we cover recent methods to analyze and model network data accessible as a stream of edges, such as interactions in a social network service, or any other graph database with real-time updates from a stream. First we introduce the data streaming computational model and give examples of the so-called temporal networks. We describe how traditional graph properties (sampling, subgraph counting, graph query evaluation, etc.), low-rank approximation, network embedding, link prediction, and centrality algorithms can be implemented and updated while the edge stream is processed. As an outlook, we discuss among others distributed data stream processing engines and concept drift detection in streams. For most part, we provide sample data and implementation as Python codes packaged in a Docker image.

## KEYWORDS

temporal networks, data stream processing

**ACM Reference Format:**
András Benczúr, Ferenc Béres, Domokos Kelen and Róbert Pálovics. 2021. Tutorial on Graph Stream Analytics. In *The 15th ACM International Conference on Distributed and Event-based Systems (DEBS '21), June 28-July 2, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3465480.3468293

## 1 INTRODUCTION

In this short tutorial, we discuss methods to analyze data streams of temporal graphs [19]. The main topic of the tutorial involves data stream processing [43] and its application to temporal networks [19]. Applications include the analysis of Twitter [60], cryptocurrency [12] and sensor network [21] data, as well as tree and graph search queries in streaming data [57], the streaming version of retrieving graphs quickly from a large database via graph-based indices [65].

We showcase the relevance of the edge steaming model for social media and networked sensor data, and experiment with sample Python implementations of selected algorithms. In this tutorial, we assume some level of familiarity with algorithms both for data streaming and for network analysis. Knowledge of word embedding methods such as the skip-gram model [46] is an advantage. For the sample program codes, knowledge of the Python programming language and the Jupyter Notebook environment[1] is needed.

The tutorial is organized around the data streaming model, the notion and examples of an edge stream, and specific algorithms for processing edge streams, in the order of the next Sections. As an overview and instructions to install all experiments as a packaged Docker image, visit the GitHub page of the tutorial[2].

## 2 THE DATA STREAMING COMPUTATIONAL MODEL

A few years ago, the term *fast data* [39] arose to capture the idea that *streams* of data are generated at very high rates from network measurements, call records, web page visits, sensor readings, financial applications [12, 70], network monitoring [1, 7], security, sensor networks [21], Twitter analysis [10, 13, 60], and more [22]. Traditional data processing assumes that data is available for multiple access, even if in some cases it resides on disk and can only be processed in larger chunks. In this case, the data is *at rest*, and we can perform *batch processing*. Database systems, for example, store large collections of data and allow users to initiate queries and transactions. Fast data, or *data in motion* is closely connected to and in certain cases used as a synonym of the *data stream* computational model [6, 48]. In this model, data arrives continuously in a potentially infinite stream that has to be processed by a resource-constrained system. The fact that only a small portion of the data can be kept available for immediate analysis [29] has both algorithmic and statistical consequences for machine learning. Suboptimal decisions on earlier parts of the data may be difficult to unwind, and if needed, require low memory sampling and summarization procedures. Many of the usual data processing operations would need random access to the data [6]. For example, only a subset of SQL queries can be served from the data stream. As surveyed in [48], data stream algorithms can tackle this constraint by a variety of strategies, including adaptive sampling in sliding windows, selecting representative distinct elements, and summarizing data in low-memory data structures, also known as sketches or synopses.

For graphs, there are two data streaming models, the *adjacency stream* model where the graph is presented as a sequence of edges in arbitrary order and there is no bound on the degree of a vertex, and the *incidence stream* model where graphs are of bounded degree and all edges incident to a vertex are presented successively [8]. In the tutorial, we consider the adjacency stream model.

*The research was supported by the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program.

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of a national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.
*DEBS '21, June 28- July 02, 2021, Milan, Italy*
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8555-8/21/06…$15.00
https://doi.org/10.1145/3465480.3468293

[1]https://jupyter.org/
[2]https://github.com/ferencberes/DEBS-graph-stream-tutorial/

## 3 EXAMPLES OF TEMPORAL NETWORKS AND EDGE STREAMS

Most of the networks in nature, society, and technology change over time. In graph theory terminology, nodes and edges get additional temporal characteristics and form a *temporal network* [19]. A large variety of temporal network algorithms have appeared for connectivity, spanning trees, matchings, and many more, which are surveyed, for example, in [2, 31].

The usual approach for analyzing temporal graphs is to create a series of snapshots, and track dynamics for various parameters in these static graphs [37, 54, 59]. However, for high temporal granularity networks, the frequent execution of graph mining algorithms on the recent snapshots could cause a significant running time overhead. Instead, high temporal granularity networks are considered in the *edge* or *graph stream* model [44] where edges must be processed once they arrive in the stream. Algorithms designed for graph streams usually have only a limited possibility to store past data [14], thus they must be *online updateable*.

Social interactions as temporal networks can be considered as edge streams [60]. We can collect all "retweets" on Twitter with corresponding hashtags to track popularity of a political party during the election period to estimate the popularity [5, 25]. Similarly, the network of cryptocurrency transactions can be analyzed for assessing anonymity [12]. Other applications include tree and graph search queries in streaming data [57], the streaming version of retrieving graphs quickly from a large database via graph-based indices [65].

As a hands-on exercise, we demonstrate how the Twitter API [10, 13, 60] can be deployed as a graph stream source that provides a "retweet" and an "@-mention" edge stream. A key issue for temporal network analysis is the difficulty of evaluation. For the evaluation of link prediction, we only require the set of edges that arrive at a given time, but other tasks such as importance or centrality rely on external ground truth labels, which often require tedious human effort even for a static network analysis. In a dynamic graph, depending on time granularity, the same human data curation may be required in each time step. As one example that we consider in this tutorial, we provide external ground truth labels for a tennis tournament Twitter collection based on the tournament schedule as an external source [10, 11].

## 4 BASIC GRAPH ALGORITHMS

Certain graph statistics such as degree have natural temporal variants applicable in the edge stream model. For example, in [34] several temporal tasks such as degree, closeness, betweenness are listed; however, they give algorithms only for static graph snapshots. Note that time-decayed degree of a set of nodes is relative straightforward to maintain in an edge stream [11].

Graph streaming algorithms that arise in database queries include finding the node with maximum degree, largest connected component, and node pair connected by the larges number of paths [30]. An essential yet highly nontrivial streaming graph property computation task is triangle counting [18, 58], a key step in computing the so-called clustering coefficient [42] of the network. Another, general direction towards handling edge streams is graph sampling [3], which provides a general way to weight edge sampling to accomplish various estimation goals of graph properties such as subgraph counts. Finding and online updating frequent graph patterns can be based on direct indexing of the most frequent subgraphs [57] or path indexing [65].

## 5 LINK PREDICTION

A common task in network analysis is the prediction of future links [41]. In a streaming setting this translates to the online link prediction task where the model should forecast the next edge that will appear in the stream. We consider two variants of this problem. In *node specific* link prediction, the goal is to predict the next edge adjacent to a predefined node. And in the *general* task, the next edge in the stream must be predicted.

To train and evaluate a time-sensitive or online link prediction method, we can use the predictive sequential, abbreviated as *prequential* method [20] in the following steps: (1) Based on the next unlabeled instance in the stream, we cast a node specific, or a general prediction, usually a ranked list of next edge candidates; (2) As soon as the actual next edge becomes available, we evaluate the prediction; (3) We update the model with the new edge before proceeding with the next one.

We showcase streaming link prediction by Alpenglow[3], a streaming recommender system research prototyping tool [23]. Note that user specific link prediction can be considered a special case of recommendation where both the "users" who enter the recommendation service and the "items" that we recommend for the users are nodes of the graph, and an edge is a recommendation of another node for a given node.

## 6 REPRESENTATION LEARNING ON GRAPHS

Embedding methods on graphs encode the nodes of the network to vectors in a low-dimensional vector space. In general, representations in the embedded space should reflect the structure of the original graph. Perhaps the most popular embedding method, adjacency matrix factorization [36] or SVD [68], can be applied for several tasks [38] such as visualization [35], clustering [17] and link prediction [45]. Incremental algorithms applicable in a streaming setting are known both for SVD [68] and for stochastic gradient descent matrix factorization [50].

Recently, random walk-based static embedding approaches have been proposed, like Node2Vec [27], LINE [62], and DeepWalk [52]. These methods sample node pairs that co-occur in random walks, and then optimize for their similarity in the embedded space, motivated by the skip-gram model from natural language processing [46]. We briefly review the methodology of the above approaches by following [28]. Static embedding methods learn an embedding vector $q_u$ for each node $u$ in the graph. Usually the objective is to learn vectors that are similar for neighboring nodes. Let $s(u)$ denote the neighborhood of $u$; then our goal is to satisfy $q_v \approx q_u$ for $v \in s(u)$. Shallow embedding approaches for static graphs differ in the objective function they use to ensure the similarity of the embeddings, and in the definition of the network neighborhood $s(u)$. Random walk-based approaches [27, 52] sample vertices from the neighborhood of a node. Sampling is done by initiating random

[3]https://github.com/rpalovics/Alpenglow

walks from node $u$. These methods optimize for cross-entropy loss:

$$\sum_u \sum_{v \in s^*(u)} = -\log \left[ \frac{\exp(q_u q_v)}{\sum_w \exp(q_u q_w)} \right]; \qquad (1)$$

where $s^*(u)$ is a random sample from the neighborhood of $u$. Most algorithms use the Word2Vec [46] model as an underlying abstraction by training the model, using sampled walks analogously to sentences to build an embedding model [53] as the vector space representation of the graph.

Partial or full online incremental updates are proposed for several variants of the walk-based embedding methods [10, 40, 49, 52, 66, 69]. We showcase the applications of graph stream embedding for Twitter [10] and cryptocurrency transaction [12] network analysis using the source code available on Github[4]. As a promising direction for computing the embedding dynamically, we mention recurrent neural networks, for example, Long Short-Term Memory networks [64], although we note that their applicability for embedding graphs is not yet explored.

## 7 GRAPH CENTRALITY MEASURES

To quantify the importance of a node, several graph centrality measures have been proposed [15]. The definitions of centrality vary greatly and incorporate both global and local factors of a node's location within the network. For temporal networks, a few generalizations of static centrality measures to dynamic settings have been suggested recently [4, 11, 26, 34, 55, 61, 63]. In these works, tracking centrality of a single node and determining its variability play a major role [63], as it has been observed in the literature that centrality of nodes can change drastically from one time period to another [16]. With the exceptions of [11, 55] and the degree in [34], the above results cannot be used for computing and updating centrality online.

We demonstrate how the generalizations of PageRank and the Katz-index for edge streams [11, 55] can be applied in social network analysis. Both of these centrality metrics build upon the concept of time respecting paths, in which adjacent edges must be ordered in time. We build on the online available Twitter centrality experiments[5] of [11].

## 8 OUTLOOK

We briefly cover data stream processing systems, concept drift detection, online training of neural networks [33] and other topics related to machine learning in big data streams [9]. Apache Spark [67] and Apache Flink [43], have the most active development for learning from streams. Other systems usually provide machine learning functionalities by interfacing with SparkML, a Spark-based machine learning library, or SAMOA [47], a stream learning library designed to work as a layer on top of general DSPEs.

Data streaming is not just a technical restriction on machine learning. Fast data is not just about processing power but also about fast semantics. Large databases available for mining today have been gathered over months or years, and the underlying processes generating them have changed during this time, sometimes radically [32]. In data analysis tasks, fundamental properties of the data may change quickly, which makes gradual manual model adjustment procedures inefficient and even infeasible [71]. Traditional, batch learners build static models from finite, static, identically distributed data sets. By contrast, stream learners need to build models that evolve over time. Processing will strongly depend on the order of examples generated from a continuous, non-stationary flow of data. Modeling is hence affected by potential concept drifts or changes in distribution [24].

[4]https://github.com/ferencberes/online-node2vec
[5]https://github.com/ferencberes/online-centrality

## SHORT BIO OF THE SPEAKERS

AB received his Ph.D. in 1997 at the Massachusetts Institute of Technology. He is leading a Data Science research laboratory of the Institute for Computer Science and Control, Hungary. He is the scientific director of the Artificial Intelligence National Laboratory, a consortium of 10 institutions in Hungary founded in 2020. The tutorial is prepared together with his doctoral students FB, DK, and RP. Since the completion of his doctoral studies, RP is postdoctoral fellow at Stanford University.

AB, DK and RP kept a hands-on Tutorial on Open Source Online Learning Recommenders [51] at RecSys 2017. AB and RP together with Levente Kocsis authored a survey on online machine learning in big data streams [9] that appeared as chapters of the Encyclopedia of Big Data Technologies [56].

## REFERENCES

[1] Daniel J Abadi, Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. 2003. Aurora: a new model and architecture for data stream management. *The VLDB Journal—The International Journal on Very Large Data Bases* 12, 2 (2003), 120–139.

[2] Charu Aggarwal and Karthik Subbian. 2014. Evolutionary network analysis: A survey. *ACM Computing Surveys (CSUR)* 47, 1 (2014), 10.

[3] Nesreen K Ahmed, Nick Duffield, Theodore Willke, and Ryan A Rossi. 2017. On sampling from massive graph streams. *arXiv preprint arXiv:1703.02625* (2017).

[4] Ahmad Alsayed and Desmond J Higham. 2015. Betweenness in time dependent networks. *Chaos, Solitons & Fractals* 72 (2015), 35–48.

[5] Pablo Aragón, Karolin Eva Kappler, Andreas Kaltenbrunner, David Laniado, and Yana Volkovich. 2013. Communication dynamics in twitter during political campaigns: The case of the 2011 Spanish national election. *Policy & Internet* 5, 2 (2013), 183–206.

[6] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. 2002. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 1–16.

[7] Shivnath Babu and Jennifer Widom. 2001. Continuous queries over data streams. *ACM Sigmod Record* 30, 3 (2001), 109–120.

[8] Ziv Bar-Yossef, Ravi Kumar, and D Sivakumar. 2002. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *SODA*, Vol. 2. 623–632.

[9] András A Benczúr, Levente Kocsis, and Róbert Pálovics. 2018. Online Machine Learning in Big Data Streams. *arXiv preprint arXiv:1802.05872* (2018).

[10] Ferenc Béres, Domokos M. Kelen, Róbert Pálovics, and András A Benczúr. 2019. Node embeddings in dynamic graphs. *Applied Network Science* 4, 64 (2019), 25.

[11] Ferenc Béres, Róbert Pálovics, Anna Oláh, and András A Benczúr. 2018. Temporal walk based centrality metric for graph streams. *Applied Network Science* 3, 32 (2018), 26.

[12] Ferenc Béres, István András Seres, András A Benczúr, and Mikerah Quintyne-Collins. 2020. Blockchain is Watching You: Profiling and Deanonymizing Ethereum Users. *arXiv preprint arXiv:2005.14051* (2020).

[13] Albert Bifet and Eibe Frank. 2010. Sentiment knowledge discovery in twitter streaming data. In *International conference on discovery science*. Springer, 1–15.

[14] Albert Bifet, Richard Kirkby, and B Pfahringer. 2011. *Data stream mining: a practical approach*. Technical Report. University of Waikato.

[15] Paolo Boldi and Sebastiano Vigna. 2014. Axioms for centrality. *Internet Mathematics* 10, 3-4 (2014), 222–262.

[16] Dan Braha and Yaneer Bar-Yam. 2006. From centrality to temporary fame: Dynamic centrality in complex networks. *Complexity* 12, 2 (2006), 59–63.

[17] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. 2016. Recent advances in graph partitioning. *Algorithm engineering*

(2016), 117–158.

[18] Luciana S Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. 2006. Counting triangles in data streams. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 253–262.

[19] Hassan Nazeer Chaudhry. 2019. FlowGraph: Distributed temporal pattern detection over dynamically evolving graphs. In *Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems*. 272–275.

[20] A Philip Dawid. 1984. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society. Series A (General)* (1984), 278–292.

[21] Gianmarco De Francisci Morales, Albert Bifet, Latifur Khan, Joao Gama, and Wei Fan. 2016. Iot big data stream mining. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2119–2120.

[22] Óscar Fontenla-Romero, Bertha Guijarro-Berdiñas, David Martinez-Rego, Beatriz Pérez-Sánchez, and Diego Peteiro-Barral. 2013. Online machine learning. *Efficiency and Scalability Methods for Computational Intellect* 27 (2013).

[23] Erzsébet Frigó, Róbert Pálovics, Domokos Kelen, Levente Kocsis, and András Benczúr. 2017. Alpenglow: Open source recommender framework with time-aware learning and evaluation. (2017).

[24] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. 2013. On evaluating stream learning algorithms. *Machine learning* 90, 3 (2013), 317–346.

[25] Daniel Gayo-Avello. 2013. A meta-analysis of state-of-the-art electoral prediction from Twitter data. *Social Science Computer Review* (2013).

[26] Peter Grindrod and Desmond J Higham. 2014. A dynamical systems view of network centrality. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, Vol. 470.

[27] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[28] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[29] Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. 1998. Computing on data streams. *External memory algorithms* 50 (1998), 107–118.

[30] Monika R. Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. 1999. Computing on data streams. In *External Memory Algorithms, DIMACS Book Series vol. 50.* American Mathematical Society, 107–118.

[31] Petter Holme and Jari Saramäki. 2012. Temporal networks. *Physics reports* 519, 3 (2012), 97–125.

[32] Geoff Hulten, Laurie Spencer, and Pedro Domingos. 2001. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 97–106.

[33] Lakhmi C Jain, Manjeevan Seera, Chee Peng Lim, and Pagavathigounder Balasubramaniam. 2014. A review of online learning in supervised neural networks. *Neural computing and applications* 25, 3 (2014), 491–509.

[34] Hyoungshick Kim and Ross Anderson. 2012. Temporal node centrality in complex networks. *Physical Review E* 85, 2 (2012), 026107.

[35] Yehuda Koren. 2003. On spectral graph drawing. In *International Computing and Combinatorics Conference*. Springer, 496–508.

[36] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[37] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. 2010. Structure and evolution of online social networks. In *Link mining: models, algorithms, and applications*. Springer, 337–357.

[38] Jérôme Kunegis, Stephan Schmidt, Andreas Lommatzsch, Jürgen Lerner, Ernesto W De Luca, and Sahin Albayrak. 2010. Spectral analysis of signed graphs for clustering, prediction and visualization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 559–570.

[39] Wang Lam, Lu Liu, STS Prasad, Anand Rajaraman, Zoheb Vacheri, and AnHai Doan. 2012. Muppet: MapReduce-style processing of fast data. *Proceedings of the VLDB Endowment* 5, 12 (2012), 1814–1825.

[40] J. B. Lee, G. Nguyen, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim. 2020. Dynamic Node Embeddings From Edge Streams. *IEEE Transactions on Emerging Topics in Computational Intelligence* (2020), 1–16.

[41] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology* 58, 7 (2007), 1019–1031.

[42] R Duncan Luce and Albert D Perry. 1949. A method of matrix analysis of group structure. *Psychometrika* 14, 2 (1949), 95–116.

[43] Volker Markl. 2018. Mosaics in Big Data: Stratosphere, Apache Flink, and Beyond. In *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems*. 7–13.

[44] Andrew McGregor. 2014. Graph stream algorithms: a survey. *ACM SIGMOD Record* 43, 1 (2014), 9–20.

[45] Aditya Krishna Menon and Charles Elkan. 2011. Link prediction via matrix factorization. In *Joint european conference on machine learning and knowledge discovery in databases*. Springer, 437–452.

[46] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.

[47] Gianmarco De Francisci Morales and Albert Bifet. [n.d.]. SAMOA: scalable advanced massive online analysis. *Journal of Machine Learning Research* 16, 1 ([n. d.]).

[48] Shanmugavelayutham Muthukrishnan et al. 2005. Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science* 1, 2 (2005), 117–236.

[49] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *3rd International Workshop on Learning Representations for Big Networks*.

[50] Róbert Pálovics, András A Benczúr, Levente Kocsis, Tamás Kiss, and Erzsébet Frigó. 2014. Exploiting temporal influence in online recommendation. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 273–280.

[51] Róbert Pálovics, Domokos Kelen, and András A Benczúr. 2017. Tutorial on Open Source Online Learning Recommenders. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 400–401.

[52] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[53] Ofir Press and Lior Wolf. 2016. Using the Output Embedding to Improve Language Models. *CoRR* abs/1608.05859 (2016).

[54] Martin Rosvall and Carl T Bergstrom. 2010. Mapping change in large networks. *PloS one* 5, 1 (2010).

[55] Polina Rozenshtein and Aristides Gionis. 2016. Temporal pagerank. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 674–689.

[56] Sherif Sakr and Albert Y Zomaya. 2019. *Encyclopedia of big data technologies*. Springer International Publishing.

[57] Dennis Shasha, Jason TL Wang, and Rosalba Giugno. 2002. Algorithmics and applications of tree and graph searching. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 39–52.

[58] Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. 2017. Triest: Counting local and global triangles in fully dynamic streams with fixed memory size. *ACM TKDD* 11, 4 (2017), 1–50.

[59] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. 2007. GraphScope: Parameter-free Mining of Large Time-evolving Graphs. In *Proceedings of the 13th International Conference on Knowledge Discovery and Data Mining (KDD)*.

[60] Abhijit Suprem and Calton Pu. 2019. Assed: A framework for identifying physical events through adaptive social sensor data filtering. In *Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems*. 115–126.

[61] John Tang, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Vincenzo Nicosia. 2010. Analysing information flows and key mediators through temporal centrality metrics. In *Proceedings of the 3rd Workshop on Social Network Systems*. ACM, 3.

[62] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1067–1077.

[63] Dane Taylor, Sean A Myers, Aaron Clauset, Mason A Porter, and Peter J Mucha. 2017. Eigenvector-based centrality measures for temporal networks. *Multiscale Modeling & Simulation* 15, 1 (2017), 537–574.

[64] Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. A unified tagging solution: Bidirectional LSTM recurrent neural network with word embedding. *arXiv preprint arXiv:1511.00215* (2015).

[65] Xifeng Yan, Philip S Yu, and Jiawei Han. 2004. Graph indexing: A frequent structure-based approach. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. 335–346.

[66] Yanwei Yu, Huaxiu Yao, Hongjian Wang, Xianfeng Tang, and Zhenhui Li. 2018. Representation Learning for Large-Scale Dynamic Networks. In *International Conference on Database Systems for Advanced Applications*. Springer, 526–541.

[67] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster computing with working sets. *HotCloud* 10, 10-10 (2010), 95.

[68] Ziwei Zhang, Peng Cui, Jian Pei, Xiao Wang, and Wenwu Zhu. 2018. Timers: Error-bounded svd restart on dynamic networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[69] D. Zhu, P. Cui, Z. Zhang, J. Pei, and W. Zhu. 2018. High-Order Proximity Preserved Embedding for Dynamic Networks. *IEEE Transactions on Knowledge and Data Engineering* 30, 11 (2018), 2134–2144. https://doi.org/10.1109/TKDE.2018.2822283

[70] Yunyue Zhu and Dennis Shasha. 2002. Statstream: Statistical monitoring of thousands of data streams in real time. In *Proceedings of the 28th international conference on Very Large Data Bases*. VLDB Endowment, 358–369.

[71] Indre Žliobaite, Albert Bifet, Mohamed Gaber, Bogdan Gabrys, Joao Gama, Leandro Minku, and Katarzyna Musial. 2012. Next challenges for adaptive learning systems. *ACM SIGKDD Explorations Newsletter* 14, 1 (2012), 48–55.