Visualizing Graph Differences from Social Media Streams

Minjeong Shin[1,2] Dongwoo Kim[1,2] Jae Hee Lee[1] Umanga Bista[1,2] Lexing Xie[1,2]
Australian National University[1], Data to Decision CRC[2]
{minjeong.shin,dongwoo.kim,jaehee.lee,umanga.bista,lexing.xie}@anu.edu.au

ABSTRACT

We propose KGdiff, a new interactive visualization tool for social media content focusing on entities and relationships. The core component is a layout algorithm that highlights the differences between two graphs. We apply this algorithm on knowledge graphs consisting of named entities and their relations extracted from text streams over different time periods. The visualization system provides additional information such as the volume and frequency ranking of entities and allows users to select which parts of the graph to visualize interactively. On Twitter and news article collections, KGdiff allows users to compare different data subsets. Results of such comparisons often reveal topical or geographical changes in a discussion. More broadly, graph differences are useful for a wide range of relational data comparison tasks, such as comparing social interaction graphs, identifying changes in user behavior, or discovering differences in graphs from distinct sources, geography, or political stance.

CCS CONCEPTS

• **Human-centered computing → Social networks**; **Visual analytics**; • **Networks** → *Online social networks*;

KEYWORDS

Relation extraction; Relational data comparison; Graph diff

ACM Reference Format:
Minjeong Shin, Dongwoo Kim, Jae Hee Lee, Umanga Bista, Lexing Xie. 2019. Visualizing Graph Differences from Social Media Streams. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), Feb. 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3289600.3290616

1 INTRODUCTION

Everyone can be a content producer and publisher on the social web. As a result, textual content is being posted at an unprecedented rate. Twitter, for instance, produces 500 million tweets per day. One efficient way to understand the vast amount of information over time is to compare differences between two different time ranges as programmers `diff` their code to understand changes.

There have been a few attempts to provide a visual comparison of two sets of documents. Wordcloud comparison [2] allows people to scan large numbers of documents in terms of word usage. Visualizing the topological difference between two biological networks [8] helps understand molecular interactions between genes. Extracting graphs from text has been an active research area [9, 10], but to the best of our knowledge, comparing graphs from different text collections has not been done.

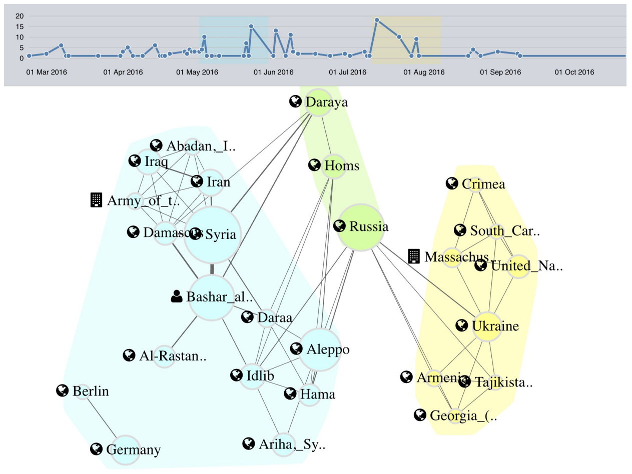


Figure 1: Comparing two subgraphs from different time ranges. The entity co-occurrence graph is extracted from tweets by @JulianRoepcke, a German journalist who actively discusses international politics. From the graph, we can observe that the discussion focus of the journalist has been changed from Syria and Aleppo in May (blue) to Ukraine in August (yellow), while Daraya, Homs (cities in Syria) and Russia are discussed (green) in both time ranges.

The text-as-graph view naturally raises three key questions. First, how can we formalize the unstructured content into a graph? To structuralize plain text, we need to extract key objects and their relations. Second, how can we diff graphs given multiple time ranges? Unlike text diff, it is unclear how to diff multiple graph structures. Third, how can we design an interactive visualization system around graph diff that provides users with additional context and content details to understand the differences?

The goal of this work is to construct a novel tool for obtaining insights from ongoing social events and news – using graph representations of online text, and provide visualizations for change tracking and highlighting over long periods of time, reducing information overload. To this end, we introduce an interactive visualization system named knowledge graph diff (KGdiff). We first introduce two representative approaches to extract entity relation graphs from plain text (section 2). We then formalize the differences between two graphs using Venn diagram (section 3). For a better visualization of Venn diagram, we add two additional conditions to the force-directed graph layout algorithms. Finally, we develop an interactive visualization system which renders the differences between two graphs dynamically according to user input (section 4). Figure 1 shows an example of graph comparison.

2 GRAPH CONSTRUCTION

In this section, we describe two graph extraction methods from text: an entity co-occurrence graph and an entity relation graph.

2.1 Entity co-occurrence graph

Named entities are real world objects such as persons, places, or organizations. For example, from the given text "Einstein was born in Germany and studied at the UZH.", we can extract three named entities as follow.

- Einstein ⟶ `Albert_Einstein (Person)`
- Germany ⟶ `Germany (Place)`
- UZH ⟶ `University_of_Zurich (Organization)`

Note that texts are presented on the left and the canonical name of entities are presented on the right.

We use co-occurrence between named entities to construct an entity co-occurrence graph. Each node in the graph corresponds to a named entity, and the edge between two nodes indicates that the two entities are used in the same sentence. To extract entities from a collection of documents, we use an entity extraction tool called DBpedia Spotlight[3]. DBpedia Spotlight takes unstructured text and extracts entities which have a corresponding page in Wikipedia. DBpedia Spotlight additionally annotates the type of an entity such as *person* or *place*. We display the three common types, *person*, *place* and *organization* and mark other types as *other*.

2.2 Entity relation graph

The co-occurrence graph shows entities that appear together, but fails to show how two entities are related. For example, given two named entities `Albert_Einstein` and `Germany`, what would be the relation between these two entities?

We introduce an entity relation graph, which visualizes the exact relations between entities, to overcome such limitation. Unlike the edges in the co-occurrence graph, the edges in the relation graph have a direction based on its relation. For example, the fact that `Albert_Einstein` was born in `Germany` can be represented as two nodes, Albert Einstein as the subject node and Germany as the object node, and the directed edge from Einstein to Germany labeled by `is_born_in`.

We use a distance supervision (DS) method [9] to extract relations between entities. DS builds a multi-class classifier trained on external knowledge bases. The training procedure takes a set of sentences containing a pair of named entities as an input. The model then looks up an external knowledge base where the relations between the pair of entities are partially known, which is used as an output of the multi-class classification model. In other words, if the knowledge base contains the triple (entity1, entity2, relation) then the sentences containing those two entities are treated as a positive example for the corresponding relation.

There has been rich literature on DS methods, and most recently, with neural networks [5]. For this work, we combine the bidirectional-RNN architecture and the attention mechanism [6], which shows an outstanding empirical performance against other neural network architecture [1].

The co-occurrence graph and relation graph complement each other. The relation graph has a less number of entity pairs than co-occurrence graph because the DS requires a predefined set of relations to train and not every entity pairs in the entity co-occurrence graph belongs to one of these relations. Therefore, the co-occurrence graph shows interactions within a larger set of entities than those of relation graph while the relation graph shows the exact relations with limited pairs of entities from DS.

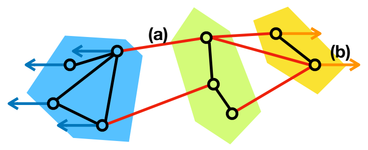


Figure 2: Venn diagram as a tool for constrained graph layout. Different node groups are represented by different convex hulls. (a) Edge length inequality constraint - those that are in different groups (red) are longer than those in the same group (black). (b) Repulsive forces applied to the central node (by degree centrality) in each group encourages two groups far away from each other.

3 GRAPH DIFF

Given two graphs, we first find the intersection and differences between the two sets of nodes. We adopt the Venn diagram to visualize those three groups, where the common nodes group is located in the middle of two exclusive node groups. Figure 2 illustrates the conceptual Venn diagram structure of two graphs, where convex hulls emphasize the different node groups. Alternative, we may compare differences between two sets of edges. In this work, we focus on comparing the differences between two sets of nodes.

Positioning nodes in two-dimensional space with respect to their group structure requires an graph layout algorithm. To visualize the graph Venn diagram, we develop a new graph layout algorithm based on the force-directed algorithm [7]. The force-directed layout uses a physics based simulator. The algorithm simulates the movements of nodes in two-dimensional space by applying two different forces simultaneously: attractive forces between connected nodes and repulsive forces between all pairs of nodes. The attractive force makes the connected nodes close to each other, while the repulsive force prevents the connected nodes from collapsing. To find an optimal position of nodes, the force-directed layout uses an iterative approach. On each turn, the nodes are re-positioned based on the two forces, and the repositioning is repeated until the positions are stabilized. The vanilla force-directed layout algorithm, however, is not aware of node groups identified in Venn diagram. We design two conditions to ensure that the layout algorithm to place different node groups apart from each other while maintaining a visible structure of nodes within each group.

Edge length constraint: We add inequality constraints that all edges between different node groups are longer than edges within

[1]Code and datasets are provided at https://github.com/dongwookim-ml/deep-distant-supervision

the same node group. The edge length constraint makes different groups visually distinguishable against each other. This constraint is illustrated in Figure 2 (a), where all cross-group edges (red) are longer than edges within group (black).

Extra repulsive force: We apply extra repulsive forces to move the two exclusive node groups in opposite directions. Repulsive forces are applied to the nodes in the exclusive groups in every iteration of the simulation. We select the highest degree node from each group and push these nodes away until the distance between the nodes reaches a certain threshold. The threshold is proportional to the total number of nodes. The extra repulsive forces are illustrated in Figure 2 (b), where the repulsive force moves the left and right groups in opposite directions.

We use the physical simulator used for the force-directed algorithm to simulate the position of nodes with additional constraint and force. The simulator runs iteratively until the node positions reach an equilibrium state.

4 KGDIFF

We describe the complete KGDIFF system[2] that supports user inquiry and interaction around graph diff, including choosing subgraphs and additional statistics in entity volumes and ranking over time. We provide the implementation details at the end.

4.1 Property panel

The property panel (Figure 3 (a)) controls what to show in the main visualization areas. There are four configurable options in the panel. The first option controls which dataset to visualize. The system keeps a set of pre-defined datasets where the entities and relations are already extracted. A user can choose one of the pre-defined data or upload a new dataset. In the latter case, the system generates entity graphs on the fly. The second option controls which connected components to visualize. Depending on the dataset, the system may extract a massive graph consisting of multiple connected components. If visualizing the whole graph is infeasible, a user can choose which connected component to show. The third option allows a user to decide the number of nodes to plot. Selected number of nodes are displayed according to their degrees (from highest to lowest). The final option allows a user to filter nodes by their types, such as person and place. Only selected node types are displayed in the entity graph area.

4.2 Interactive graph panel

Interactive graph panel consists of two sub-panels, a timeline chart and an entity graph.

Timeline chart (Figure 3 (b)) shows the volume of entities over time, where the x-axis represents time slots equally divided by a certain amount of time, and the y-axis represents the total number of entities extracted in the given time slot.

Entity graph (Figure 3 (c)) visualizes entity graphs and their differences. Entity nodes are represented with their names. Node color with a prefix icon is used to indicate its type. The size of a node indicates its degree. The entity graph can have either co-occurrence edges or relation edges according to graph types. Co-occurrence edges are undirected and do not have edge label. The thickness of each edge is used to show the number of occurrences. Relation edges use the relationship between connected nodes as a label.

The timeline chart shows the overall trend of the data based on the number of entities. Entities graphs are generated by selecting time slots from the timeline chart. One can control the length of time slot using the drop down list on the top-right. Each time slot is click-able and, by clicking time slot, a user can draw an entity graph extracted from the selected time slot. Selecting multiple time slots enables the system to generate an entity graph from the union.

Graph diff functionality is enabled when a user double-clicks one or more time slots. The selected time slots are presented in a different color and are considered as the second time ranges. The system automatically diffs two graphs from the selected time ranges via the graph differencing algorithm.

4.3 Entity trend panel

While the entity graphs and their diff help understand the differences between two time ranges, KGDIFF also visualizes entity trend over time according to their volume changes to help users understand the general trend in a corpus. To present the volume changes of entities, we design two visualization methods, an entity stream and an entity ranking chart.

Entity stream (Figure 3 (d)) summarizes the volume changes of the top 20 entities over time. The x-axis aligns with the timeline chart, and the y-axis represents the volume of entities. Unlike the timeline chart where the volume represents the total number of entities, the entity stream focuses on top 20 entities. Top 20 entities are stacked from bottom to top, where the volume of an individual entity is distinguished by its color scheme represented on the right side.

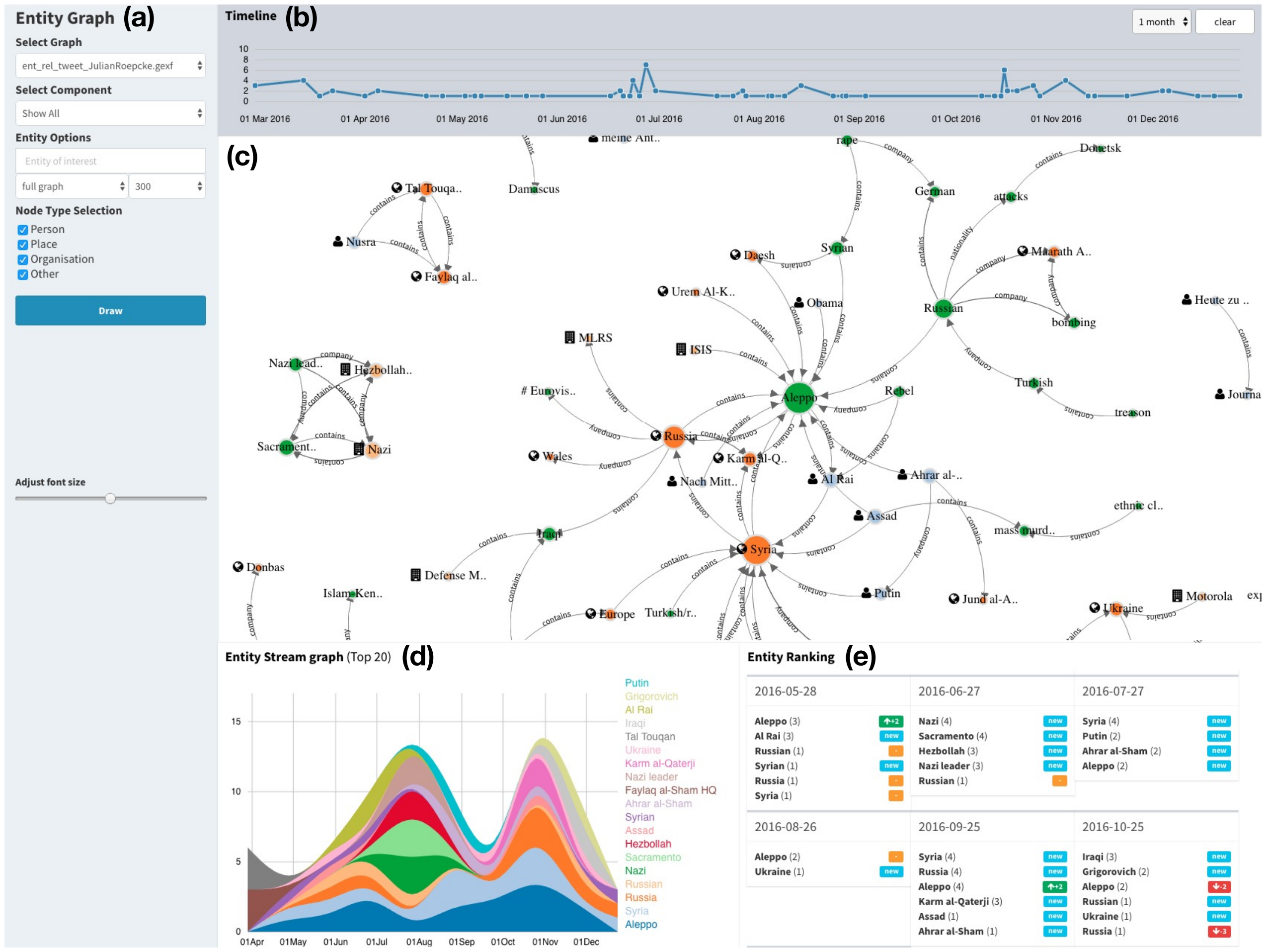
Entity ranking chart (Figure 3 (e)) consists of ranking boxes, and each ranking box contains the top 20 entities for each time slot. We adapt the idea of a weekly music chart to show changes in entity ranking measured by the number of appearances in a given time slot. The length of each time slot is aligned with the timeline chart and the stream graph. Up and down-arrows next to the entities are used to show the difference in the entity ranking compared to the previous time slot. If an entity is first introduced in a given time range, 'new' indicator is used to represents the new entity.

4.4 Implementation details

KGDIFF is implemented in Python with the open-source web framework Django. We use the NetworkX [4] Python library for the graph creation and manipulation. D3 toolkit [1] is used for plotting the timeline chart, entity stream, and entity graph. The web user interface is powered by HTML and JavaScript and polished using Bootstrap and AdminLTE template.

KGDIFF uses a graph file format internally to maintain the graph structure with additional information such as timestamps on edges and types of nodes. All common text formats with timestamps can be used to generate a graph. To create a graph structure from a text, we use NLTK to parse sentences. All pairs of entities extracted from a sentence become nodes and an edge with a timestamp connects them. Generated graphs are stored in file formats that NetworkX enables storage and reloading of the files.

[2]A video of the KGDIFF is provided at https://goo.gl/UWmQgU



Entity Graph

Select Graph: ent_rel_tweet_JulianRoepcke.gexf

Select Component: Show All

Entity Options: Entity of interest; full graph; 300

Node Type Selection: ☑ Person ☑ Place ☑ Organisation ☑ Other

Adjust font size

Timeline: 1 month

Entity Stream graph (Top 20)

Entity Ranking

2016-05-28		2016-06-27		2016-07-27	
Aleppo (3)	↑+2	Nazi (4)	new	Syria (4)	new
Al Rai (3)	new	Sacramento (4)	new	Putin (2)	new
Russian (1)	-	Hezbollah (3)	new	Ahrar al-Sham (2)	new
Syrian (1)	new	Nazi leader (3)	new	Aleppo (2)	new
Russia (1)	-	Russian (1)	-		
Syria (1)	-				

2016-08-26		2016-09-25		2016-10-25	
Aleppo (2)	-	Syria (4)	new	Iraqi (3)	new
Ukraine (1)	new	Russia (4)	new	Grigorovich (2)	new
		Aleppo (4)	↑+2	Aleppo (2)	↓-2
		Karm al-Qaterji (3)	new	Russian (1)	new
		Assad (1)	new	Ukraine (1)	new
		Ahrar al-Sham (1)	new	Russia (1)	↓-3

Figure 3: Snapshot of KGDIFF system presenting the entire entity relation graph extracted from tweets by @JulianRoepcke. The system consists of (a) Property panel, (b) Timeline chart, (c) Entity graph, (d) Entity stream, and (e) Entity ranking chart.

5 CONCLUSION

In this paper, we construct entity graphs to extract structured information from free text. We develop the KGDIFF system for visualizing differences between two graphs interactively, with user input and providing additional data profiles.

The system can be applied to other types of graphs to visualize differences between two graphs, although we have only applied KGDIFF on tweet stream in this work. Future work include comparing social interaction graphs and identifying changes in user behavior. We will also provide the functionality to compare different sets of edges to visualize the topological changes of graphs.

Acknowledgements. This research was supported by use of the NeCTAR Research Cloud, a collaborative Australian research platform supported by the National Collaborative Research Infrastructure Strategy.

REFERENCES

[1] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (Dec. 2011), 2301–2309. https://doi.org/10.1109/TVCG.2011.185

[2] Quim Castella and Charles Sutton. 2014. Word storms: Multiples of word clouds for visual comparison of documents. In *Proceedings of the 23rd international conference on WWW*. ACM, 665–676.

[3] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving Efficiency and Accuracy in Multilingual Entity Extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*.

[4] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*. 11 – 15.

[5] Shantanu Kumar. 2017. A Survey of Deep Learning Methods for Relation Extraction. *arXiv preprint arXiv:1705.03645* (2017).

[6] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the ACL*, Vol. 1. 2124–2133.

[7] Thomas M. J. Fruchterman and Edward Reingold. 1991. Graph Drawing by Force-Directed Placement. 21 (11 1991), 1129–1164.

[8] Raghvendra Mall, Luigi Cerulo, Halima Bensmail, Antonio Iavarone, and Michele Ceccarelli. 2017. Detection of statistically significant network changes in complex biological networks. In *BMC Systems Biology*.

[9] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th IJCNLP*. ACL, 1003–1011.

[10] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the ACL*, Vol. 2. 207–212.