Who killed Lilly Kane? A case study in applying knowledge graphs to crime fiction.

Mariam Alaverdian
Department of Mathematics
Los Angeles City College
Los Angeles, CA 90029
masha.alaverdyan@gmail.com

William Gilroy
Department of Mathematics
Harvey Mudd College
Claremont, CA
wgilroy@hmc.edu

Veronica Kirgios
Department of Mathematics
Notre Dame Univ.
Notre Dame, IN 46556
vkirgios@nd.edu

Xia Li
Department of Mathematics
University of California, Los Angeles
Los Angeles, USA 90095
xli51@math.ucla.edu

Carolina Matuk
Department of Mathematics
Univ. of Iowa
Iowa City, Iowa 52242-1419
carolinamatuk@hotmail.com

Daniel McKenzie
Department of Mathematics
University of California, Los Angeles
Los Angeles, USA 90095
mckenzie@math.ucla.edu

Tachin Ruangkriengsin
Department of Mathematics
University of California, Los Angeles
Los Angeles, USA 90095
bankcrub@g.ucla.edu

Andrea L. Bertozzi
Department of Mathematics
University of California, Los Angeles
Los Angeles, USA 90095
bertozzi@ucla.edu

P. Jeffrey Brantingham
Department of Anthropology
University of California, Los Angeles
Los Angeles, USA 90095
branting@ucla.edu

***Abstract*—We present a preliminary study of a knowledge graph created from season one of the television show *Veronica Mars*, which follows the eponymous young private investigator as she attempts to solve the murder of her best friend Lilly Kane. We discuss various techniques for mining the knowledge graph for clues and potential suspects. We also discuss best practice for collaboratively constructing knowledge graphs from television shows.**

I. Introduction

Knowledge graphs are a powerful tool for organizing, storing and presenting complex data. A knowledge graph is a graph of data whose nodes represent entities of interest and whose edges represent relations between these entities. Formally, a knowledge graph consists of a set of entities, $\mathcal{V}$, a set of relations or predicates, $\mathcal{R}$ and a set of facts, $\mathcal{E}$, which specify pairwise relations between entities. Crucially, the facts must obey rules specified by an accompanying ontology, $\mathcal{O}$, which dictates which kinds of relations can be present between which kinds of entities. For a comprehensive and modern introduction to knowledge graphs, we refer the reader to [HBC$^+$20].

A. Knowledge graphs for fiction

Despite the promise that knowledge graphs hold in analyzing semantic data, they have not yet been extensively applied in analyzing fiction. We believe that knowledge graphs are a promising tool for encoding and analyzing the complex human-human interactions present in novels, movies and television shows, and further that these domains form a realistic proxy for real-world human-human interactions. Inspired by a knowledge graph challenge problem [KET$^+$19], we chose to study the genre of crime fiction. Unlike the Sherlock Holmes novel studied in [KET$^+$19], we chose to focus on a television show, *Veronica Mars*. While there has been prior work on analyzing movies and television shows using graph theory [BDE$^+$16], [BEGM18], we believe that we are the first to apply *knowledge graphs* to television. Television has at least two distinct advantages over novels:

- T.V. scripts are more structured than novels. In principle this makes it easier to automate knowledge graph construction.
- Continuity over episodes and seasons allows for the construction of a larger and richer knowledge graph.

B. Veronica Mars

Set in a fictional town in California, *Veronica Mars* is a modern day spin on *Nancy Drew*. The teenage protagonist, Veronica Mars, is a private investigator. The series begins with the murder of Veronica's best friend, Lilly Kane. A minor character, Abel Koontz, is convicted of her muder but there is reason to suspect that he is not the true culprit. This search for Lilly's true murderer forms the overarching plot motif. Within each episode Veronica is presented with a crime, gathers evidence and (usually) solves the case. Unlike

We are grateful for support from UCLA and Harvey Mudd College as well as the Los Angeles City College STEM Pathways program, supported by Department of Education PR# P031C160251. This work was also partially supported by NSF grants DMS-1737770 and DMS-2027277.

popular detective fiction such as Sherlock Holmes, the facts of the case are usually transparently presented throughout the episode and the identity of the culprit can usually be deduced straightforwardly from these facts, without recourse to *deus ex machina*.

C. Notation

We shall denote our knowledge graph as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$. Entities and relations shall be written in typewriter font, for example `VeronicaMars` [1]. Facts shall be written as triples of the form (`Entity1`,`Relation`,`Entity2`). Occasionally (see Section II-C), we shall suppose partial knowledge of $\mathcal{G}$, and in particular assume that there are true relations between entities that are not captured by $\mathcal{E}$. We shall refer to these unknown triples as unknown or missing facts. We shall denote by G the underlying undirected graph of $\mathcal{G}$.

We experimented with various software for constructing knowledge graphs, for example Karma [KSA+12]. However we found that our data was too unstructured for most such tools. The process of constructing a knowledge graph from *unstructured* data is less well studied, although we note the recent works [LZL+18], [KI17]. In particular, Seq2RDF [LZL+18] is a powerful tool for turning sentences into triples that agree with a given ontology. In principle such a tool could be used to automate the construction of a knowledge graph from the script of a T.V. show. However, we found several roadblocks to this approach:

- State-of-the-art tools such as Seq2RDF are currently only able to extract one triple per sentence, and these triples need to be of a fairly quotidian nature (for example "Berlin is the capital of Germany"). They are not able to handle compound sentences, synonyms or metaphors which are common in fiction.
- Some important facts are conveyed over multiple, non-contiguous sentences, conveyed visually or implied without ever being explicitly stated.

Thus, we chose to manually construct the knowledge graph. Our workflow was as follows: two or three team members would watch an episode and then collaboratively identify which facts presented in the episode were important. These were then captured in a spreadsheet. The spreadsheets for each episode were then combined, and the Python package RDF lib was used to convert them into a collection of RDF triples. Further details are presented in Sections I-D and I-E. Our resulting knowledge graph contains 541 entities and 1,106 facts. The complete data set, as well as all Python code used in constructing it, is available at [AGK+20].

D. The ontology and allowed relations

Our ontology was based on the "friend of a friend" ontology, and thus allows for common relations between characters (e.g. "friend of", "child of"). The ontology automatically encodes the fact that some relations are symmetric (if "A is friend of B" then also "B is friend of A") or have obvious inverses (if "A is child of B" then "B is parent of A"). We augmented this ontology to allow for relations between characters particular to crime fiction ("A is kidnapped by B") as well as relevant relations between characters and places, objects and abstract concepts such as financial status. We took a flexible approach and added new relations as they occurred in the show. A representative portion of our ontology is displayed as Figure 1.

E. Which facts are important?

The process of choosing which facts in a given episode to record and add to the knowledge graph is somewhat subjective. We focused on facts that:

- Captured important biographical information of characters.
- Captured important relationships between characters.
- May be clues, both for the episode specific case and the overarching case.

For example, the case in episode six centers on a rigged student council election, yielding triples such as (`Wanda_Varner`, `runs_for`, `student_council`) and (`Madison_Sinclair`, `threatens`, `Wanda_Varner`). However, key clues to the overarching case (*i.e.* the murder of Lilly Kane) are also revealed in this episode, yielding: (`white_sneakers`, `clue_of`, `Case1`) and (`white_sneakers`, `seen_at`, `Lilly_Kane's_room`). More examples may be found in Table I.

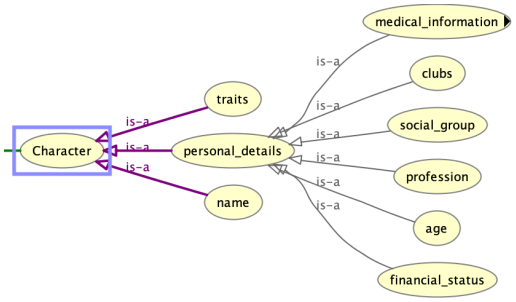

Fig. 1. A snapshot of hierarchical relations in the ontology.

Subject	Predicate	Object
`Veronica_Mars`	`child_of`	`Keith_Mars`
`Weevil_Navarro`	`has_financial_status`	`lower_class`
`Wallace_Fennel`	`in_club`	`basketball`
`Don_Lamb`	`employee_of`	`Keith_Mars`
`Van_Clemmons`	`has_last_name`	`Clemmons`

TABLE I
SOME CHARACTER-SPECIFIC FACTS IN THE *Veronica Mars* KNOWLEDGE GRAPH

F. Reification

In analyzing crime fiction, recording the time at which certain facts are revealed is essential. To do this we applied reification, a technique originally used for reifying an

[1] and note the distinction between the fictional person, Veronica Mars, the television show, *Veronica Mars*, and the entity in our knowledge graph `Veronica_Mars`!

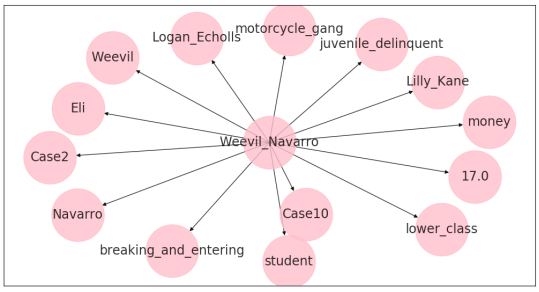

Fig. 2. The character subgraph for Weevil Navarro, with temporal labels suppressed.

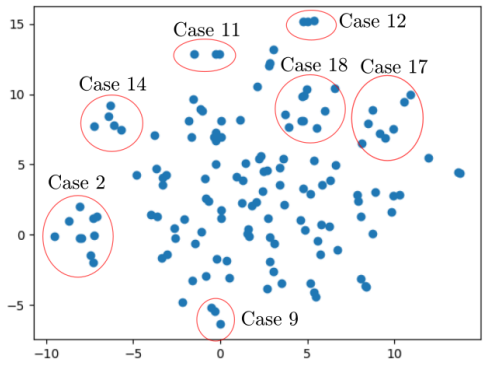

Fig. 3. 2D TSNE representation of the `TransE` embedding. Here we have represented the embeddings of clues from various cases. The red circles indicate that clues which are from the same case, in other words semantically similar entities, seem close to each other in the TransE emebedding. For example, in the cluster of case 11, the entities consist of the `asphyxiation` (the means of the crimes), `Vic Sciaraffa` (the character), `videotapes`, `number` and `wristband` (the clues).

attributed knowledge graph, to record temporal information. In the process of reification, relations are first "reified" into entities. These entities can then be the subject of a fact. We then add the relation `occurs_at` to $\mathcal{R}$. A time stamp can then be added by including the fact (`Reified_relation`,`occurs_at`, time_stamp) in $\mathcal{G}$.

We note that reification could also be used to record the *provenance* of facts. For example, if the fact "character-A was seen at location-B" is revealed by character-C, it could be useful to record this. Thus, if at a later stage character-C is revealed to be unreliable, one can easily search for the facts that may no longer be valid.

G. Character subgraphs

Because the knowledge graph is constructed using information from every episode, one can easily query it to produce longitudinal subgraphs. For example, one can extract the subgraph of all triples containing a particular character as a subject. This provides a snapshot of a character's story arc through the series, and also identifies the cases they were involved in. See, for example, Figure 2.

II. Analysis

With our knowledge graph in hand, we attempted to use it to answer several questions:

1) Can we identify important clues for a given case?
2) Can we extract overarching themes or topics from the knowledge graph that might not be apparent given only individual episodes?
3) Can we identify who killed Lilly Kane?

In this section we present preliminary results on all three of these problems.

A. Identifying relevant clues

We used TransE [BUGD+13] to embed our knowledge graph into $\mathbb{R}^{200}$. Recall that TransE assigns every entity $v \in \mathcal{V}$ to a vector $\mathbf{u}_v \in \mathbb{R}^{200}$ *and* every relation $r \in \mathcal{R}$ to a vector $\mathbf{u}_r \in \mathbb{R}^{200}$ such that, for every fact $(v_1, r, v_2) \in \mathcal{E}$ we have that $\mathbf{u}_{v_1} + \mathbf{u}_r \approx \mathbf{u}_{v_2}$. Specifically, TransE aims to find an embedding minimizing the loss:

$$\sum_{(v_1,r,v_2)\in\mathcal{E}} \|\mathbf{u}_{v_1} + \mathbf{u}_r - \mathbf{u}_{v_2}\|_2 - \sum_{(v_1,r,v_2)\notin\mathcal{E}} \|\mathbf{u}_{v_1} + \mathbf{u}_r - \mathbf{u}_{v_2}\|_2$$

TransE is designed to place semantically similar entities and relations together, so we hypothesized that it would place relevant clues close to the case with which they were associated. To visualize this embedding, we used t-SNE [MH08] to project from $\mathbb{R}^{200}$ to $\mathbb{R}^2$ (see [AGK+20] for details). The result of this projection is shown in Figure 3. Even after this enormous reduction in dimension, we observe some encouraging results. For at least 7 of the 19 cases, multiple relevant clues for these cases are indeed grouped together (shown circled in red).

B. Topic extraction using random walks

Given a corpus $\{d_1, \ldots, d_n\}$ of documents, the process of extracting relevant topics from this corpus is well studied:

1) Identify m keywords relevant to this corpus.
2) For each d_i construct a vector $\mathbf{v}_i$ of (possibly weighted) counts of keywords.
3) Form the matrix $X = \begin{bmatrix}\mathbf{v}_1 & \ldots & \mathbf{v}_n\end{bmatrix}$.
4) Compute a (low-rank) non-negative matrix factorization: $X \approx UV$ with $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{r\times n}$
5) Obtain r topics, one for each column of U.

The implicit assumption that makes this procedure work is that the number of topics is much less than the number of documents, and that each document is a superposition of a small number of topics.

It is not clear how to adapt this to analyze a season of a television show. The key issue is how to "slice" the season into documents. The naive solution of declaring each episode to be a document is unsatisfactory because:

- There are relatively few episodes in a season.

- A given story arc is often developed over multiple episodes. For example, in *Veronica Mars* details relating to Lilly Kane's murder are presented to the viewer in the form of flashbacks interspersed throughout season one.

Inspired by [LNB19], [LMS19] we propose a novel approach:

1) Fix a required number of documents, n.
2) Let G denote the undirected graph obtained by forgetting the orientation and type of each relation in $\mathcal{G}$.
3) For $i = 1, \ldots, n$, choose a random initial vertex $v_0^{(i)}$ and perform a ℓ step random walk starting from $v_0^{(i)}$. Let:
$$d_i = \left(v_0^{(i)}, e_1^{(i)}, v_1^{(i)}, \ldots, v_\ell^{(i)}\right)$$
be the collection of vertices and edges traversed by this walk.
4) Using $d_1, \ldots, d_n$ our documents, perform the topic modeling as described earlier.

Note that using this "motif sampling", one can easily generate a large corpus from a single season. More importantly, we hypothesize that these randomly sampled "motifs" are more likely to capture important topics than episodes, as they will contain related elements that span multiple episodes. We applied the technique to the *Veronica Mars* knowledge graph with $n = 1000$ and $l = 50$. We used TF-IDF to vectorize the resulting documents, and performed NMF with $r = 25$. The reason we use TF-IDF here is to balance out the importance of nodes—as nearly every character is related to Veronica, it is very likely that every random walk will contain `Veronica_Mars`. While the resulting topics were somewhat noisy, the results are encouraging. For example, Topic 12 contains the entities `Aaron_Echolls`, `tapes` `affair_with` and `Lilly_Kane`. This topic neatly summarizes the circumstances of Lilly Kane's death: Aaron Echolls killed Lilly Kane in a fit of rage after video tapes documenting their affair came to light. Other topics relate to important characters (Topic 13) or to episode-specific cases (Topic 20). The complete list of topics is available at [AGK+20].

Topic 12	Topic 13	Topic 20
Aaron_Echolls	Duncan_Kane	Wanda_Varner
Lilly_Kane	blackout	rigged_election
affair_with	epilepsy	ballot_instructions
tapes	oxcarbazepine	Madison_Sinclair

TABLE II
SELECTED TOPICS

C. Link Prediction

The link prediction problem takes two entities, v_1 and v_2, and a relation $r \in \mathcal{R}$ and asks whether the triple (v_1, r, v_2) should be a fact in the knowledge graph $\mathcal{G}$. Ideally, one would like to use link prediction to deduce the guilty parties in the various cases solved by Veronica Mars by taking $v_1 =$`Character_A`, $r =$`described_as` and $v_2 =$ `perpetrator`. Due to the lack of training data (there are only 20 cases), we found this challenging. Thus, we also investigated using link prediction to determine whether or not two characters were friends by choosing $v_1 =$`Character_A`, $v_2 =$ `Character_B` and $r =$`friend_of`.

Many approaches to link prediction first construct a vector embedding of $\mathcal{G}$ and then assign a probability to (v_1, r, v_2) being a fact inversely proportional to $\|\mathbf{u}_1 + \mathbf{u}_r - \mathbf{u}_{v_2}\|_2$ (see [RFM+20] for an overview). We experimented with using TransE for link prediction in this manner, but found the results to be unsatisfactory. We hypothesize that this is because the complex social links we are seeking to predict are more appropriately captured by a subgraph than by an embedding. For example, the data of a crime could be represented by a subgraph connecting the perpetrator, the victim, a motive for the crime, a location of the crime and several damning pieces of evidence.

Motivated by this hypothesis, we investigated link prediction algorithms that are subgraph based. In particular, we used SEAL [ZC18]. Given a putative triple (v_1, r, v_2), SEAL constructs an enclosing subgraph around this link and then uses a trained graph neural network (GNN) to output a probability of this link being a true fact. It is important to note that SEAL is designed for undirected graphs, and thus is not cognizant of the type or direction of the relation r. SEAL also does not use the ontology in any way. Like any GNN, SEAL requires training in the form of positive examples of the link we are trying to predict. Negative examples are generated by random sampling. We experimented with 50%, 75% and 90% of our data as training data (with the rest held back as a test set).

We found mixed results. For example, using a 90% split SEAL assigned an encouraging 71% probability to the triple $($`Aaron_Echolls`, `described_as`, `perpetrator`$)$, correctly identifying Aaron Echolls as a suspect even though this was not given in the training data. Unfortunately it assigned similarly high probabilities to false triples such as $($`Logan_Echolls`, `described_as`, `perpetrator`$)$. We suspect that the poor performance of SEAL here can be explained by the paucity of training data—our knowledge graph has only 541 vertices, whilst the examples considered in [ZC18] all have at least 10,000. Moreover, SEAL ignores the rich information encoded in the *types* of relations, treating them all uniformly as edges. A recent work [TDH19] extends SEAL to handle multiple kinds of relations and might yield better results.

III. Conclusions and future directions

In this paper we considered the novel problem of studying television series using knowledge graphs. We introduced a new knowledge graph data set, which may be of interest to the community. We proposed several novel analysis techniques, such as random walk topic modelling, and tested the applicability of existing techniques to this new domain. We believe that

applying knowledge graphs to fiction has tremendous scope, and for future groups we offer the following recommendations:

- Before adding any facts to the knowledge graph, develop an application-specific ontology allowing for fewer relations. When developing the knowledge graph, only add additional relations if strictly necessary. This will mitigate a problem that we encountered, namely rare relations that only occur once or twice in the knowledge graph.
- While it is tempting to use natural language processing to automate the process of extracting facts, we found these tools unable to deal with the linguistic complexity of fiction. Hence, we recommend manually extracting facts.
- We recommend using time stamps. This can be done either using reification (as we have done) or by using an attributed knowledge graph.
- The proposed random walk topic modelling scheme seems very promising. An interesting question for future groups would be to compute statistics on coverage (*i.e.* how many entities are included in at least one document) and repetition (*i.e.* how many documents the average entity appears in). This could assist in selecting the number of documents to generate from a given a knowledge graph. Another interesting line of research would be to make the random walk *ontology aware*. For example, if the edge traversed at step i represents a certain kind of relation, this information could be used to restrict the relations which the edges at step $i + 1$ can represent. It seems likely that incorporating such additional structural information into the random walks will yield more coherent documents.
- We suspect that GNN approaches may play an increasingly important role in link prediction; further studies might consider starting their work with GRAIL [TDH19].
- Template matching [MCT+18], [KET+19] is promising strategy for identifying meaningful subgraphs within the knowledge graph. We note that this approach would benefit from a more principled ontology with fewer relations, as discussed above.

REFERENCES

[AGK+20] Mariam Alaverdian, William Gilroy, Veronica Kirgios, Xia Li, Carolina Matuk, Daniel Mckenzie, and Tachin Ruangkriengsin. https://github.com/DanielMckenzie/KnowledgeGraphs_REU2020, 2020.

[BDE+16] Anthony Bonato, David Ryan D'Angelo, Ethan R Elenberg, David F Gleich, and Yangyang Hou. Mining and modeling character networks. In *International workshop on algorithms and models for the web-graph*, pages 100–114. Springer, 2016.

[BEGM18] Anthony Bonato, Nicole Eikmeier, David F Gleich, and Rehan Malik. Dynamic competition networks: detecting alliances and leaders. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 115–144. Springer, 2018.

[BUGD+13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[HBC+20] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Jose Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. Knowledge graphs. *arXiv preprint arXiv:2003.02320*, 2020.

[KET+19] Takahiro Kawamura, Shusaku Egami, Koutarou Tamura, Yasunori Hokazono, Takanori Ugai, Yusuke Koyanagi, Fumihito Nishino, Seiji Okajima, Katsuhiko Murakami, Kunihiko Takamatsu, et al. Report on the first knowledge graph reasoning challenge 2018. In *Joint International Semantic Technology Conference*, pages 18–34. Springer, 2019.

[KI17] Natthawut Kertkeidkachorn and Ryutaro Ichise. T2kg: An end-to-end system for creating knowledge graph from unstructured text. In *AAAI Workshops*, 2017.

[KSA+12] Craig A Knoblock, Pedro Szekely, José Luis Ambite, Aman Goel, Shubham Gupta, Kristina Lerman, Maria Muslea, Mohsen Taheriyan, and Parag Mallick. Semi-automatically mapping structured sources into the semantic web. In *Extended Semantic Web Conference*, pages 375–390. Springer, 2012.

[LMS19] Hanbaek Lyu, Facundo Memoli, and David Sivakoff. Sampling random graph homomorphisms and applications to network data analysis. *arXiv preprint arXiv:1910.09483*, 2019.

[LNB19] Hanbaek Lyu, Deana Needell, and Laura Balzano. Online matrix factorization for Markovian data and applications to network dictionary learning. *arXiv preprint arXiv:1911.01931*, 2019.

[LZL+18] Yue Liu, Tongtao Zhang, Zhicheng Liang, Heng Ji, and Deborah L McGuinness. Seq2rdf: An end-to-end application for deriving triples from natural language text. *arXiv preprint arXiv:1807.01763*, 2018.

[MCT+18] Jacob D Moorman, Qinyi Chen, Thomas K Tu, Zachary M Boyd, and Andrea L Bertozzi. Filtering methods for subgraph matching on multiplex networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3980–3985. IEEE, 2018.

[MH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[RFM+20] Andrea Rossi, Donatella Firmani, Antonio Matinata, Paolo Merialdo, and Denilson Barbosa. Knowledge graph embedding for link prediction: A comparative analysis. *arXiv preprint arXiv:2002.00819*, 2020.

[TDH19] Komal K Teru, Etienne Denis, and William L Hamilton. Inductive relation prediction by subgraph reasoning. *arXiv*, pages arXiv–1911, 2019.

[ZC18] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pages 5165–5175, 2018.