Ginkgo-P: General Illustrations of Knowledge Graphs for Openness as a Platform

Blaine Hill
blaineh2@illinois.edu
Department of Computer Science,
University of Illinois
Urbana-Champaign
Urbana, Illinois, USA

Lihui Liu
lihuil2@illinois.edu
Department of Computer Science,
University of Illinois
Urbana-Champaign
Urbana, Illinois, USA

Hanghang Tong
htong@illinois.edu
Department of Computer Science,
University of Illinois
Urbana-Champaign
Urbana, Illinois, USA

ABSTRACT

Accessibility and openness are two of the most important factors in motivating AI and Web research. One example is as costs to train and deploy large Knowledge Graph (KG) systems increases, valuable auxiliary features such as visualization, explainability, and automation are often overlooked, diminishing impact and popularity. Furthermore, current KG research has undergone a vicissitude to become convoluted and abstract, dissuading collaboration. To this end, we present Ginkgo-P, a platform to automatically illustrate any KG algorithm with nothing but a script and a data file. Additionally, Ginkgo-P elucidates modern KG research on the UMLS dataset with interactive demonstrations on four categories: KG Node Recommendation, KG Completion, KG Question Answering, and KG Reinforcement Learning. These categories and their many applications are increasingly ubiquitous yet lack both introductory and advanced resources to accelerate interest and contributions: with just a few clicks, our demonstration addresses this by providing an open platform for users to integrate individual KG algorithms. The source code for Ginkgo-P is available: we hope that it will propel future KG systems to become more accessible as an open source project.

CCS CONCEPTS

• **Computing methodologies** → **Reasoning about belief and knowledge**; • **Information systems** → **Data mining**.

KEYWORDS

Knowledge Graph Reasoning, Knowledge Graph Visualization, Knowledge Graph Systems, Knowledge Graph Accessibility

ACM Reference Format:

Blaine Hill, Lihui Liu, and Hanghang Tong. 2024. Ginkgo-P: General Illustrations of Knowledge Graphs for Openness as a Platform. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining (WSDM '24), March 4–8, 2024, Merida, Mexico.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3616855.3635701

WSDM '24, March 4–8, 2024, Merida, Mexico

ACM ISBN 979-8-4007-0371-3/24/03…$15.00
https://doi.org/10.1145/3616855.3635701

1 INTRODUCTION

Knowledge Graphs (KGs) are relational representations that contain nodes (denoting a subject or entity) and edges (denoting a verb or dependence) which connect nodes in either a uni-directional or bi-directional manner. Specifically, a KG consists of factual triplets of the form $\langle h, r, t \rangle$ representing a head node, relation edge, and tail node. Through this, mathematical symmetry/asymmetry, inversion, and composition can be represented and used in many applications.

Most research regarding Knowledge Graph Reasoning (KGR) prioritizes designing cutting-edge algorithms for *specific* domains. As such, there is a large gap where users outside these domains are unable to familiarize themselves with KGR. Furthermore, the complexity of integrating or accessing KG systems has grown in proportion with size increases of KGs themselves [3, 17]: this exacerbates the challenge of implementation for outside users.

We developed the Ginkgo-P demonstration for this reason; it allows users simple, sleek access to an open and customizable platform which illustrates general Knowledge Graph Reasoning models. Specifically, we codify seven foundational algorithms on the Unified Medical Language System (UMLS) dataset as seen in Table 1. In this manner, Ginkgo-P appeals to both the novice and expert: a novice can sample different core KG algorithms on UMLS to build intuition while an expert can seamlessly visualize any KG algorithm.

Tasks	Algorithm	Tasks	Algorithm
KG-C	TransE [2], ComplEx [23] RotatE [18], DistMult [26]	KG-QA	EmbedKGQA [15]
Node Recommendation	PageRank [1]	KG-RL	MultiHopKG [8]

Table 1: Overview of Different Algorithms in Ginkgo-P.

Our contributions are three-fold. First, we build a system for web practitioners and researchers alike which supports general Knowledge Graph Reasoning tasks. Second, we develop a platform to illustrate and compare/contrast existing algorithms, prepackaging four categories of reasoning tasks ourselves. Lastly, we deploy and expose this platform to remove even more pain points of implementation by making our system open and accessible through the crucial ability to integrate future KG algorithms. The Ginkgo-P platform will run real-time queries and show interactive results. This may occur in one of two ways: (1) a user may interact with a responsive and interactive web application to visualize several *preexisting* algorithms (which were selected with consideration of significance and complexity in mind) operating over the UMLS [5] KG; (2) a user may design his or her own *custom* KG algorithm and execute it in the same environment as our prepackaged algorithms. Lastly, Ginkgo-P is published, served, and accessible online[1].

[1] https://ginkgo-p.onrender.com/

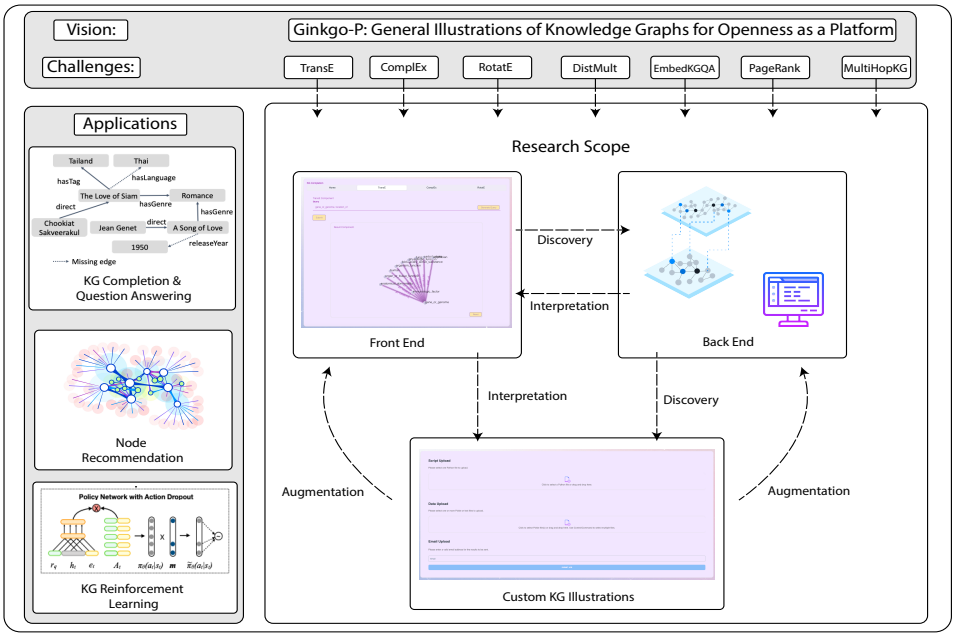

Figure 1: A diagram of the Ginkgo-P architecture.

2 RELATED WORK

Knowledge Graph Reasoning (KG-R) aims to solve different logical reasoning tasks, whereby inferring new knowledge from existing ones [12]. For example, in [2, 18, 23], the authors to predict a missing tail entity given a partial triple $\langle h, r, ? \rangle$, while in [11, 15], the authors leverage the KG to answer natural language "one-hop" or "multi-hop" input questions. In [13], the authors use the KG to verify the truthfulness of multi-modal news claims.

KG with Reinforcement Learning (KG-RL) is another way to perform KGR by modeling queries using a Markov Decision Process (MDP), learning a policy π to reach for a target node or edge, whereby inference actions are drawn from π [8, 19, 25]. KG-RL has also been used in conjunction with KG-C [25] and with KG-QA [8] to learn distant connections. KG-RL tackles not only the obstacles of KG-C or KG-QA but also the hurdle of learning optimal recollection to recall inferred relationships in a KG.

However, there exist only a few works that focus on building KGR systems, including: (1) [10], where the authors build a system to support comparative reasoning; (2) [20], where the authors construct a system used to check the truthfulness of input claims. In all of these, there is no emphasis on facilitating exploration; Ginkgo-Pboth showcases several algorithms and user input.

3 SYSTEM DESIGN

Ginkgo-P has three main architectural components branching into smaller subcomponents: (1) a navigation and interactive visualization *frontend* component which is the main interface for users; (2) a scalable, modularized KG algorithm-processing *backend* component which processes and handles "demo" requests to show examples of KG algorithms; (3) a *custom* component which handles users' "custom" requests (consisting of custom algorithm and data files). The backend component contains four branches of KG category subcomponents which encapsulate the four tasks from Table 1. A diagram of Ginkgo-P can be seen in Figure 1.

Frontend. As seen in Figure 1, users select to run their own custom KG algorithm with their own data or to run a demonstration on algorithms from Table 1. For the latter, once the appropriate query for a category is submitted, the backend retrieves and executes the query, or if a custom algorithm returns proper JSON according to Ginkgo-P's specification, a unique graph visualization component is generated. The dynamic illustration is interactive and the user's results will be highlighted as seen in Figure 2, revealing the path from the input to the result in the KG. For users who wish to provide a graph to visualize in JSON form, the request circumvents the backend and shows the interactive result immediately.

Backend. The backend mimics a RESTful API architecture [7] with a cache. Once a request from a user is received, the backend will verify the input and check the cache to see if previous users have requested similar inputs. If the results are found in the cache, they are immediately returned back to the frontend. If they aren't, the backend will spin up a child process of one of several programming languages, run the request, store it in the cache, and return the result to the frontend.

Additionally, Ginkgo-P handles parallel demonstration requests using shared memory on the KG. This design choice bypasses the restriction of spinning up many instances of the same KG to run different queries. Overall, Ginkgo-P efficiently handles any size KG (provided minimal RAM).

4 SYSTEM DETAILS

In this section, we introduce the details of the algorithms included in Ginkgo-P. For demonstration purposes, each of these models is trained and deployed using the UMLS dataset [5].

4.1 Integrating Individual Algorithms.

As introduced above, Ginkgo-P has two main features: (1) supporting multiple KG algorithms as listed in Table 1; (2) allowing users to incorporate custom algorithms. Ginkgo-P permits users to

integrate individual, custom algorithms and custom KGs, meaning that the platform is itself both model-agnostic and data-agnostic; for example, if a user wants to run a custom KG-QA algorithm on a custom KG, it it sufficient to simply upload (or reference in a script) the algorithm script and KG data file to Ginkgo-P. Yet, if a user wishes to visualize their algorithm but run it on their own system, they must adhere to JSON formatting as defined in the documentation. Interested readers may refer to our source code[2] for further details on implementation, scalability, and performance; one important note is that Ginkgo-P is constrained only by the compute resources of the hardware it is deployed from. The goal of this is to take the pain from setting up scaffolding to run one's KG research algorithms and obtain an intuitive visualization. This simplified approach enhances the user experience and eliminates unnecessary obstacles, allowing researchers to focus more on the core aspects of their KG research.

4.2 Knowledge Graph Node Recommendation

Node Recommendation is formulated as predicting a neighboring node $\hat{n}$ given a source node $s \in G$ and $\hat{n} = \min_{n \in G} d(\hat{n}, s)$ where $d(x, y)$ refers to a user-defined distance function between nodes x and y. Nominally, this is a alignment task. In practice, if the node space is not intractable, this is a simple calculation of the distance between each pair of nodes in the embedding space. If it is, we utilize the PageRank-Nibble [1] algorithm to instead search a dense cluster (e.g. a subgraph) w.r.t to the seed node. This has a large benefit of reducing the problem's asymptotic memory complexity from linear to constant. A core insight of PageRank-Nibble is to calculate an approximate PageRank vector where the running time is not impacted by the size of the input graph. This is achieved by running the Nibble partition: sweeping through the PageRank vector to identify a cut with a low conductance, producing a localized divide.

Because UMLS's PageRank scores can be pre-computed, Ginkgo-P deploys and serves regular PageRank without Nibble for the sake of simplicity [1], but PageRank-Nibble is an important algorithm for users who input large, custom KGs. Other methods like Random Walk with Restart [21, 22], (Local) Spectral Clustering [14] can also be used for the simplified Node Recommendation task. We leave them to future work.

4.3 Knowledge Graph Completion

KG-C evaluates relationships between entities in a KG using only the direct connections between entities or connections which are "one-step" or "one-hop" from the starting entity. Given a partial triple $\langle ?, r, t \rangle$, $\langle h, r, ? \rangle$ or $\langle h, ?, t \rangle$, KG-C predicts the entity or relationship most likely to complete the triple. In Ginkgo-P, we only focus on solving $\langle h, r, ? \rangle$. This reasoning is useful for swiftly predicting connections using simple questions and is used as a building block for more complex reasoning tasks. Common KG-C algorithms include TransE [2], TransH [24] and so on.

In Ginkgo-P, we implement four contrasting KGC methods: TransE [2], ComplEx [23], RotatE [18], and DistMult [26]. Their details are important and we leave them to be reviewed outside the scope of this demonstration, but are foundational to many other KG algorithms and research. Other KG-C methods such as those based on neural networks, including Graph Neural Networks (GNNs) [4, 16, 27] can also be used. We leave these to future work.

[2]https://github.com/blainehill2001/Ginkgo-P

4.4 Knowledge Graph Question Answering

In KG-QA, the goal is to answer [natural language] questions by querying a KG, locating relevant triplets, and returning an entity. Different from KG-C, KG-QA may require reasoning over one hop *or* multiple hops according to the input query. For Ginkgo-P, we implement EmbedKGQA, which contains two parts [15]. First, it represents entities and relations in the KG as vector embeddings analogously to TransE or ComplEx. Futhermore, BERT [6] is used to consume the input, learning contextualized natural language embeddings [6]. Second, it defines a scoring function to measure the similarity between candidate questions and answers. The scoring function is based on ComplEx [23] and it is given by:

$$Pr(\mathbf{e}_i | \mathbf{q}, v_h, \mathcal{G}) = Re(< \mathbf{q}, \mathbf{e}_h, \bar{\mathbf{e}}_i >) \quad (1)$$

where $\mathbf{e}_i$ is the entity in the KG, $\mathbf{q}$ is the question embedding, and $\mathbf{e}_h$ is the embedding of the topic entity in the input question.

Other KG-QA algorithms, like BiNet, a multi-task approach designed to address both KG-C and KG-QA concurrently, converge towards strong weight optimization in a joint manner [11]. PrefNet [9] improves accuracy by utilizing a re-ranking module. We plan to integrate these two algorithms into the system, either in a subsequent release or in the next version.

4.5 Knowledge Graph Reasoning with Reinforcement Learning

KG-RL is the most challenging of these tasks due to its complexity: it requires both traditional approaches in combination with long-distance reasoning. In Ginkgo-P, we use KG-RL across multiple hops; as a model operates on different time steps, it traverses the KG, following and learning multiple relationships between the branches to predict an answer. In Ginkgo-P, we implement MultiHopKG [8], a method to address KG-QA with RL. It answers an input query by modeling a MDP with the KG as the state-space (or environment) and the relationships between entities as the action-space.

KG-RL can be applied to problems in KG-QA, KG-C, and more. Representing KG tasks with MDPs opens the door to reimagining previous approaches and is a wide-open area of future research.

5 SYSTEM DEMONSTRATION

Figure 2 presents the interface of Ginkgo-P with what the user will see to run a custom experiment and two examples of included algorithms run on UMLS. For the first example, the middle image shows the results of TransE on the input query $\langle$`amino_acid_peptide_or_protein`, `causes`, ?$\rangle$. The top answers are denoted by the edges of `amino_acid_peptide_or_protein` with the best answer shown in yellow. The second image shows the results of MultiHopKG on the input query $\langle$`immunologic_factor`, `disrupts`, ?$\rangle$. As we can see, despite there being no direct edges connecting `immunologic_factor` and `body_substance`, MultiHopKG locates the result because `immunologic_factor` is connected to `body_substance` via the node `tissue`. The result, denoted by the yellow edges, is

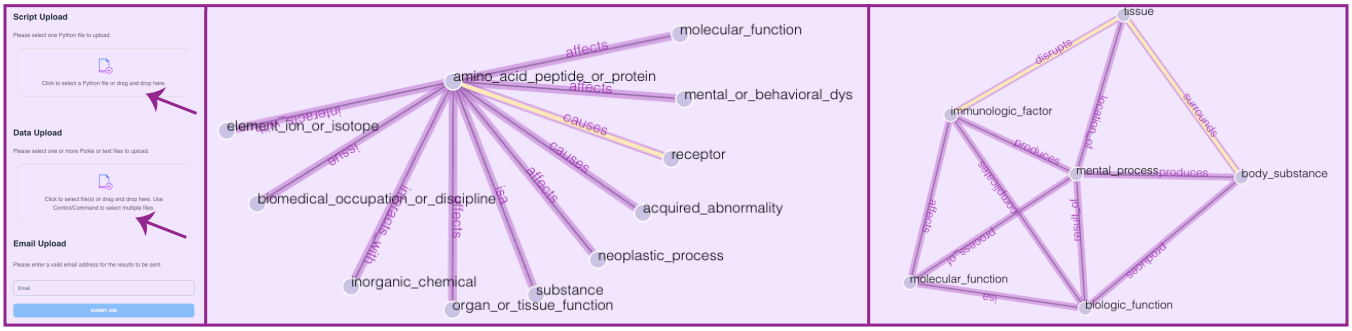

Figure 2: From left to right, one can see the Custom frontend input form, the result of TransE for the input ⟨`amino_acid_peptide_or_protein`, `causes`, ?⟩, and the result of MultiHopKG for the input ⟨`immunologic_factor`, `disrupts`, ?⟩.

[⟨`immunologic_factor`, `disrupts`, `tissue`⟩, ⟨`tissue`, `surrounds`, `body_substance`⟩].

Users of Ginkgo-P may fall into one of two categories: (1) introductory web practitioners who are interested in visualizing KG algorithms; (2) expert graph researchers who wish to implement and deploy their novel algorithms without headache. Ginkgo-Pwill be fully deployed and accessible to allow for open interaction with any audience. The goal of Ginkgo-P is to propel the audience towards future KG research directions by introducing more researchers to these core tasks and to automatically and effortlessly visualize algorithms.

6 ACKNOWLEDGEMENTS

This work is supported by NSF (1947135, 2134079, 1939725, 2316233, and 2324770), DARPA (HR001121C0165), DHS (17STQAC00001-07-00), NIFA (2020-67021-32799) and ARO (W911NF2110088).

7 CONCLUSION

In this paper, we demonstrate the first open general Knowledge Graph Reasoning platform Ginkgo-P to spur both fresh interest and future work on Knowledge Graph (KG) systems. Ginkgo-P includes several prepackaged algorithms over four categories: KG Node Recommendation, KG Completion, KG Question Answering, and KG with Reinforcement Learning. This platform allows users to visualize and compare/contrast both the prepackaged algorithms as well as any algorithms the user designs and integrates using Ginkgo-P.

REFERENCES

[1] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 475–486.
[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* 26 (2013).
[3] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. 2015. One trillion edges: Graph processing at facebook-scale. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1804–1815.
[4] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2D Knowledge Graph Embeddings. https://doi.org/10.48550/ARXIV.1707.01476
[5] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[7] Roy Thomas Fielding. 2000. *Architectural styles and the design of network-based software architectures*. University of California, Irvine.
[8] Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568* (2018).
[9] Lihui Liu, Yuzhong Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. 2023. Knowledge Graph Question Answering with Ambiguous Query. In *Proceedings of the ACM Web Conference 2023*.
[10] Lihui Liu, Boxin Du, Yi Ren Fung, Heng Ji, Jiejun Xu, and Hanghang Tong. 2021. KompaRe: A Knowledge Graph Comparative Reasoning System. In *Proceedings of the 27th ACM SIGKDD (KDD '21)*.
[11] Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. 2022. Joint Knowledge Graph Completion and Question Answering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1098–1108.
[12] Lihui Liu and Hanghang Tong. 2023. Knowledge Graph Reasoning and Its Applications. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5813–5814.
[13] Lihui Liu, Ruining Zhao, Boxin Du, Yi Ren Fung, Heng Ji, Jiejun Xu, and Hanghang Tong. 2022. Knowledge Graph Comparative Reasoning for Fact Checking: Problem Definition and Algorithms. *Data Engineering* (2022), 19.
[14] Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems* 14 (2001).
[15] Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th ACL*.
[16] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling Relational Data with Graph Convolutional Networks. https://doi.org/10.48550/ARXIV.1703.06103
[17] Nasrullah Sheikh, Xiao Qin, Berthold Reinwald, and Chuan Lei. 2022. Scaling knowledge graph embedding models. *arXiv preprint arXiv:2201.02791* (2022).
[18] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197* (2019).
[19] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
[20] Andon Tchechmedjiev, Pavlos Fafalios, and Katarina Boland. 2019. ClaimsKG: A knowledge graph of fact-checked claims. In *The Semantic Web–ISWC 2019*. Springer.
[21] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *Sixth international conference on data mining (ICDM'06)*. IEEE, 613–622.
[22] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2008. Random walk with restart: fast solutions and applications. *Knowledge and Information Systems* 14 (2008), 327–346.
[23] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*. PMLR, 2071–2080.
[24] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 28.
[25] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690* (2017).
[26] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).
[27] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. https://doi.org/10.48550/ARXIV.1412.6575