Stunning Doodle: a Tool for Joint Visualization and Analysis of Knowledge Graphs and Graph Embeddings

Antonia Ettorre, Anna Bobasheva, Franck Michel, Catherine Faron

Stunning Doodle: a Tool for Joint Visualization and Analysis of Knowledge Graphs and Graph Embeddings

Antonia Ettorre ✉[0000−0003−4868−2584], Anna Bobasheva[0000−0003−0395−2069], Franck Michel[0000−0001−9064−0463], and Catherine Faron[0000−0001−5959−5561]

Université Côte d'Azur, CNRS, Inria, I3S, Sophia Antipolis, France
{aettorre,bobasheva,fmichel,faron}@i3s.unice.fr

Abstract. In recent years, the growing application of Knowledge Graphs to new and diverse domains has created the need to make these resources accessible and understandable by users with increasingly diverse backgrounds. Visualization techniques have been widely employed as means to facilitate the exploration and comprehension of such data sources. Moreover, the emerging use of Knowledge Graph Embeddings as input features of Machine Learning methods has given even more visibility to this kind of representation, but raising the new issue of understandability and interpretability of such embeddings. In this paper, we show how visualization techniques can be used to jointly explore and interpret both Knowledge Graphs and Graph Embeddings. We present *Stunning Doodle*, a tool that enriches the classical visualization of Knowledge Graphs with additional information meant to enable the visual analysis and comprehension of Graph Embeddings. The idea is to help the user figure out the logical connection between (1) the information captured by the Graph Embeddings and (2) the structure and semantics of the Knowledge Graph from which they are generated. We detail the use of *Stunning Doodle* in a real-world scenario and we show how it has been helpful to interpret different Graph Embeddings and to choose the most suitable with respect to a specific final goal.

Keywords: Knowledge Graphs Visualization · Graph Embeddings

1 Introduction

During the last decade, the adoption of Knowledge Graphs (KGs) in multiple domains has increased steadily such that more and more projects rely on this kind of representation to store their data without compromising the semantics they bear. The fame of KGs has kept growing even more as they started to be used as information sources for many AI-powered applications in the most diverse fields, e.g. education [7, 10], medicine [19] and finance [13]. One of the reasons for their increasing success is the possibility to easily employ them as input features for several Machine Learning methods by using a low-dimensional representation of such KGs, obtained through the graph embedding process. The diversity of

circumstances in which Graph Embeddings (GEs), and therefore KGs, can be used opened the way to the exploitation of these data sources by users from various communities and with diverse backgrounds. In this context, two major needs for such users arise: *(i)* exploring and understanding the content and the structure of KGs and *(ii)* analyzing and interpreting the information captured by their GEs.

In recent years, the need for solutions to allow simple and straightforward exploration of KGs has stimulated the development of a plethora of visualization tools for such data sources. Indeed, visualization techniques are recognized as one of the main means to provide an immediate and simple understanding of complex concepts and structured data. Ideally, users could be able to gain in-depth understanding of the structure of a KG by analyzing the sheer ontologies and vocabularies on which the KG is based, and when it comes to RDF-based KGs, they could explore its content by means of SPARQL queries. Nevertheless, studying ontologies and vocabularies and writing SPARQL queries can be rather cumbersome tasks, especially when dealing with very large KGs. Moreover, these operations can only be carried out by practitioners of Semantic Web languages. Visualization techniques can help and simplify the exploration and understanding of KGs by non-expert users, offering easy-to-use interfaces, advanced statistics, and interaction functionalities. However, the visual exploration of KGs is a non-trivial task mainly due to the amount and heterogeneity of the information possibly described by KGs.

The comprehension of the information captured by GEs is an even more challenging process. Indeed, GEs are computed using "black-box" Machine Learning (ML) techniques that translate each element in the graph into a low-dimensional vector. Even though the algorithmic process to compute embeddings is well understood, a relation between the characteristics and role of the element in the graph and its vector representation in the embedding space cannot be established with certainty. In other words, multiple questions cannot be answered easily, such as:

- What do my embeddings represent?
- How are they related to the structure and semantics of my KG?
- How can I improve my embeddings to be better suited to my downstream application?

Recently, several research efforts have been made in this direction to start making sense of the information captured by GEs. Some approaches propose explainable models for computing GEs [12] or implement explanation strategies for specific embedding models [22]; while others propose methods to verify whether some specific piece of knowledge represented in a KG is actually encoded and captured by its GEs [9].

In this work, we aim to tackle this issue from a different point of view. We think that, as for KGs, the information borne by GEs could be explored and unveiled through the use of visualization techniques that would favor the discovery of the logical connection between the graph and its embeddings. Our goal is therefore to provide a visualization tool supporting the analysis and

decoding of the information captured by KGEs by unveiling the relationship between, on one hand the structure and semantics of the KG, and on the other hand the KGEs generated from it.

To achieve this goal, in this paper we present *Stunning Doodle*, a tool designed for the visualization of RDF-based KGs and GEs. *Stunning Doodle* first provides a visualization of the graph to be analyzed, offering a rich overview of both the structure and the semantics of the data. We believe that this visualization should allow users to gain a general and immediate understanding of the displayed KG while presenting a complete and detailed view of each of its components. More interestingly, this visualization is enriched by connecting the nodes in the graph with the corresponding GEs to be analyzed. It enables to select a node in a KG and visualize its neighborhood in the embedding space, and conversely to pick any of its neighbors in the embedding space and visualize their links in the KG. We argue that the joint visualization of GEs and the KG from which they are generated, with its structure and semantics, will help users, may they be non-expert users, RDF experts or Machine Learning experts, to gain interesting insights into the information captured during the embedding process.

The remainder of this paper is organized as follows. In Section 2 we present our KGEs analysis tool, while in Section 3 we demonstrate how our tool can be used and useful in two real-world use-cases. In Section 4 we review related work. Finally, conclusions and future work are discussed in Section 5.

2 Stunning Doodle

Stunning Doodle has been developed to fill the gap in the field of visual analysis of KGEs. Its main goal is to provide users with an advanced visualization of RDF graphs, enriched with information extracted from the GEs generated from those graphs. To achieve this goal, *Stunning Doodle* offers two main functionalities: (1) the visualization and navigation of RDF graphs and, (2) for each node, the enrichment of such visualization with the addition of its neighborhood in the embedding space, w.r.t. a chosen similarity measure. Additionally, *Stunning Doodle* presents semantic-based filtering and customization functionalities for nodes and links which make the visualization clearly understandable even for users with no familiarity with RDF and SPARQL concepts. To improve the readability and comprehension of the displayed KG, *Stunning Doodle* implements a partial visualization of the graph and allows users to navigate and explore it incrementally.

2.1 Knowledge Graphs Visualization

The first function of *Stunning Doodle* is the dynamic visualization of a KG using a regular graph layout. The process starts with the upload of the RDF graph file in Turtle syntax. Once the file is uploaded, the graph will be displayed as shown in Fig. 1. On the left side of the page, under the “Upload Graph” menu, three menus provide relevant information about the graph: the list of the declared

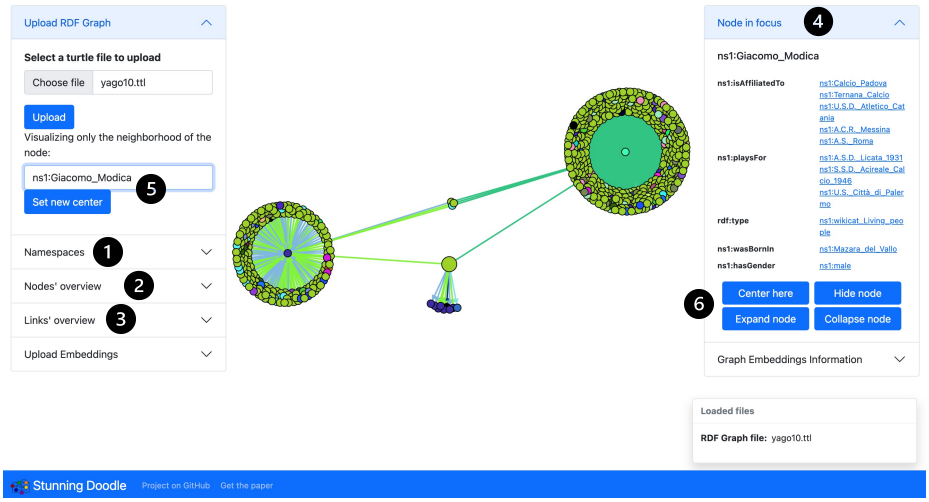

Fig. 1. Screenshot of the partial visualization of YAGO3 illustrating *Stunning Doodle*' exploration capabilities.

namespaces with their prefixes (1), and the legend, and advanced customization options for nodes (2) and links (3) that will be detailed later in this section. The visualization is interactive: nodes can be moved and zoom and spanning functionalities are available. Nodes can be selected by clicking on them, and the triples associated with the currently selected node are listed in the component "Node in focus" (4) on the right. In the example of Fig. 1, which shows an extract of YAGO3 [14], the selected node is the largest one at the bottom of the graph (`ns1:Giacomo_Modica`) and the triples for which this node is a subject are listed on the right. Each object in this list can be clicked to select the corresponding node in the graph.

Graph Exploration Interface. One of the main characteristics of *Stunning Doodle* is the ability to display nodes incrementally, thanks to the graph exploration system.

To deal with the large number of nodes possibly described in a KG, existing visualization tools employ different strategies: *(i)* relying on clusterized views which group similar (or close) nodes together [18], *(ii)* visualizing nodes incrementally based on the displayed area [3, 4], or *(iii)* showing only the nodes that are considered to be more relevant based on diverse statistical metrics [20]. In *Stunning Doodle* we opted for a different approach: a user-guided incremental exploration. Indeed, through our tool, the user decides which nodes to visualize according to their interests and needs. This choice is motivated by the fact that different users can have different, possibly very specific requirements, e.g. analyzing only the facts related to individuals of a given type while disregarding all the other possible predicates; and these requirements do not always correspond

Fig. 2. *Stunning Doodle* interface with the available customization options.

to the standard clusterization criteria or relevance metrics. As Fig. 1 shows it, after the upload of an RDF graph, only one node, randomly selected among all the nodes in the graph, is displayed alongside its close neighborhood, i.e. nodes at one-hop distance from the main node. Users can change the displayed node by typing the URI of the node they want to visualize in the component "Upload Graph" and setting it as a new center (5). Users can also navigate the graph starting from a node of interest by using the buttons in the "Node in focus" component (6). Indeed, once a node is selected, users can decide to either *(a)* center the graph on that node, causing only the selected node and its neighborhood to be visualized, while hiding the other nodes and links; *(b)* hide the node, producing the removal of that node together with the nodes that are connected only to it; *(c)* expand the node, i.e. adding the direct neighbors of the node to the visualization; and *(d)* collapse the node, i.e. removing all the nodes that are only connected to that node while keeping the node itself. For example, the visualization in Fig. 1 has been obtained by centering the graph on the node `ns1:male` at first, and then expanding the selected node (`ns1:Giacomo_Modica`) to display its neighborhood.

Filtering and Customizing Nodes and Edges. Another helpful feature of *Stunning Doodle* is the possibility of filtering nodes based on their types (values of property `rdf:type`) and customizing nodes and links colors. These features help the user to easily recognize the nodes and links they want to analyze and to focus only on them while removing information that is irrelevant for their exploration needs.

In Fig. 2 we show an example customized visualization of an RDF graph and the settings of the visualization modes. These are optional and can be activated

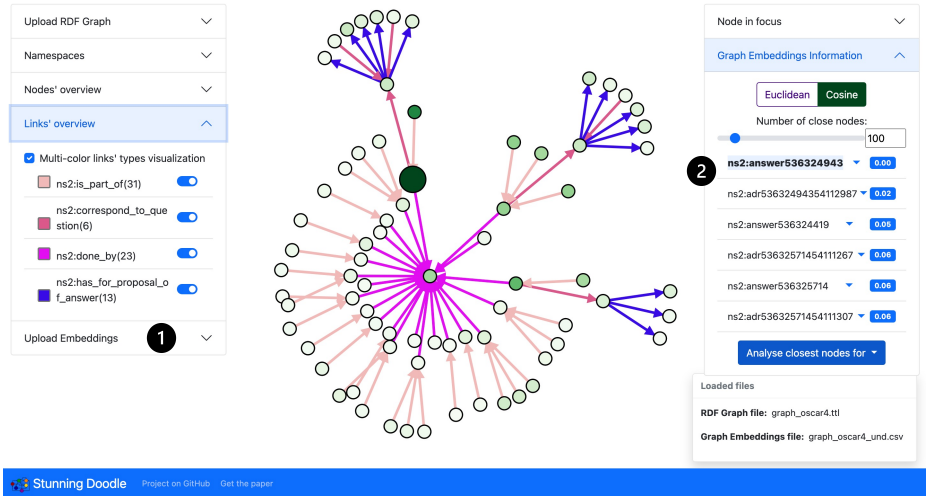

Fig. 3. Screenshot of *Stunning Doodle* showing closest nodes in the embedding space.

by selecting the corresponding options in the "Nodes' overview" and "Links' overview" menus. In the "Links' overview" menu (1), all the possible predicates are listed, each one with the number of edges of that type currently displayed on the graph and the corresponding colors which can be customized by the user (2). Similarly, in the "Nodes' overview" menu (3), the types of all the displayed nodes are listed with the count of occurrences of every type (4). For each type, a filter is added to hide/show the corresponding nodes (5). In the same component, we find two additional options that allow the user to enable and disable labels and literals visualization (6). Once activated, the former will display labels, i.e. values of properties `rdfs:label` and `skos:prefLabel`, next to the nodes subject of these properties; while the latter will add literals as leaves in the graph.

2.2 Graph Embeddings Visual Analysis

In addition to the dynamic visualization of a KG, the key functionality provided by *Stunning Doodle* which represents a major step forward when compared with the state-of-the-art tools described in Section 4 is the possibility of having a first, simple visual analysis of the GEs computed from the visualized KG.

As explained in Section 1, the recent success of KGs is mainly due to the growing number of AI applications relying on KGs through the use of KGEs. Unfortunately, the main stumbling block to extensive utilization of such representation is its difficult interpretability.

With *Stunning Doodle*, we take a first step in the understanding of KGEs through the joint visualization of both KGEs and the KG from which they are generated. More precisely, *Stunning Doodle* enables to visualize, for each node, its closest nodes in the embedding space, according to both euclidean and cosine

distance. Fig. 3 shows an example of KGEs analysis. The "Upload Embeddings" menu allows the user to upload a CSV file containing the GEs computed from the visualized KG (1). Then, the user can select a node of interest and visualize its closest neighbors in the embedding space. The closest nodes are displayed with a gradient of color that represents their distance from the node whose embedding is analyzed, i.e. darker nodes are closer (in the embedding space) to the selected node while lighter nodes are more distant. If a relation between any couple of visualized nodes exists in the KG, then the corresponding link is directly displayed in the graph according to the selected customization settings.

The list of the closest nodes with their distance is shown in the "Graph Embeddings Information" menu on the right side of the page (2). Together with this list, additional options allow the user to choose the desired distance metric (Euclidean or cosine in the current implementation), and to customize the number of closest nodes to be displayed. The example in Fig. 3 shows the 100 closest nodes in the embedding space to the node `ns1:answer536324943`, according to cosine distance. The node currently selected (biggest node) is the node that we are analyzing and from which the distances are computed. This is evident by the fact that its URI is the first item in the list of nodes and it is also the darkest node in the graph, as the distance from itself is 0. The different shades of green of each node clearly highlight which nodes among the 100 visualized are closer, while the links show how they are connected in the KG.

While visualizing a node's neighbors in the embedding space, it is still possible to access the functionalities for navigating in the KG through the buttons in "Node in focus". Therefore, any displayed node can be expanded to show additional nodes linked to it in the KG even if they are not close in the embedding space. Naturally, any new node can be further expanded to visualize the desired portion of the KG. All the links and the displayed nodes whose embeddings are not close to the initial node will be visualized according to the selected customization options in "Nodes' overview" and "Links' overview". To switch back to the simple view of the KG it is sufficient to recenter the graph on any of the displayed nodes.

To sum up, *Stunning Doodle* enables the user to understand in a glance which nodes of a KG are considered to be similar in the embedding space, while keeping track of their connections in the KG. This permits to gain immediate insights on the information captured by KGEs, e.g. which predicates have the highest impact or what connectivity patterns are more taken into account during the embedding process.

2.3 Software Design and Limitations

For the sake of simplicity and flexibility of use, *Stunning Doodle* has been implemented as a lightweight web application relying on Python and Javascript, respectively for back-end and front-end. Its setup is fairly simple as it requires only to create a Python virtual environment and it allows the easy deployment on a server to be accessed remotely by multiple users through a web browser. *Stunning Doodle* uses as input for the graph visualization a RDF Knowledge

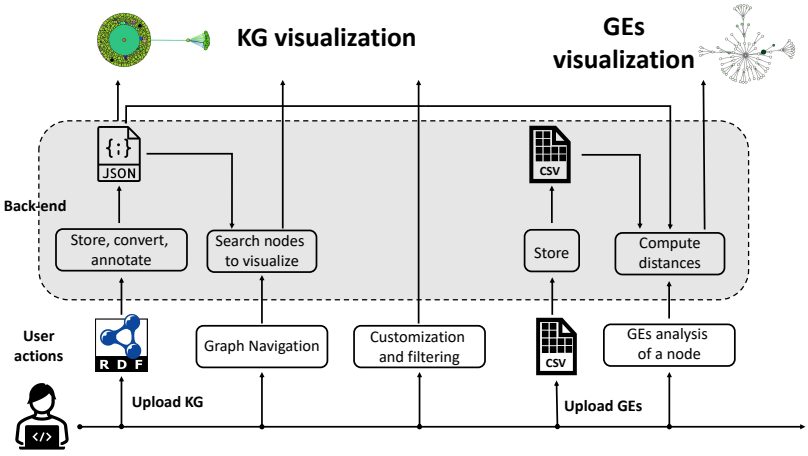

Fig. 4. Schema of the processing pipeline implemented in *Stunning Doodle*.

Graph stored in a file using Turtle syntax. For the analysis of the GEs, a CSV file containing the nodes' URIs associated with the corresponding embeddings is required. Fig. 4 illustrates the processing pipeline of *Stunning Doodle* and shows how the interactions with users are handled. Firstly, users need to upload the RDF graph they want to visualize. This graph is parsed and converted into JSON, enriching each node description with information about its types and labels to enable on-the-fly customization of the visualization based on this information. When users navigate the graph by re-centering it or expanding new nodes, a remote request to the back-end is performed to compute and retrieve the new list of nodes to visualize. Since the actions of customization and filtering concern only the nodes and edges currently visualized, they do not require any remote request to the server. To enable the functionalities of analysis of GEs, a CSV file storing such embeddings must be uploaded by the user and stored on the server. At this point, users can analyze the embeddings of any displayed node, by visualizing its closest nodes in the embedding space. When this action is executed, the back-end is queried to compute the list of the closest nodes with their distances. After mapping the closest embeddings with their corresponding nodes and computing the edges connecting them, the results are returned to the front-end that displays the obtained subgraph.

The main limitation of *Stunning Doodle* is represented by the size of the KG to be analyzed. Indeed, the larger the KG and, consequently, the embeddings file are, the longer is the time needed to upload such files, to search for the nodes to be displayed and to compute the distances between embeddings. Table 1 reports the time needed to execute the main actions provided by *Stunning Doodle* on the graph shown in Fig. 1 and Fig. 2, which contains 920.435 triples describing 124.982 nodes. These timings were obtained running *Stunning Doodle* on a MacBook Pro with 2,8 GHz Quad-Core Intel Core i7 and 16GB RAM.

Table 1. Execution time with number of displayed nodes when performing actions for the analysis of a subset of YAGO3.

	Load graph	**Set center** (`ns1:male`)	**Expand node** (`ns1:AC_Sparta_Prague`)	**Closest nodes**
Time	120s	21,7s	12,03s	5,4s
# nodes	-	502	271	100

2.4 Software Availability and Reusability

Stunning Doodle is made available as an open-source software under the MIT License. The tool is identified by means of a DOI provided through Zenodo [8] to improve its accessibility and citability. Moreover, *Stunning Doodle* source code is publicly accessible on the GitHub repository[1], which also includes extensive documentation and examples to guide the users. We plan on keeping improving *Stunning Doodle* with new functionalities and offering best-effort support to future users in case of bugs or issues of any kind. Furthermore, we welcome any feedback, idea or contribution by the community.

3 Use Cases: the OntoSIDES Scenario

To demonstrate the usefulness of *Stunning Doodle* and the added value of the functionalities it implements, we describe hereafter the two main use cases that motivated the development of such a tool. Both use cases stem from the *SIDES* real-world project in the field of e-education which aims to provide highly intelligent learning services to the medical students in France. The SIDES project involves the manipulation of OntoSIDES [17], a large KG describing the French higher education system for medical studies, with universities, students, learning material, and interactions between them. The graph includes in total more than 9.2 billion triples. To be able to exploit such an amount of information in an AI-powered downstream application, a first approach would be to rely on KGEs, as done in [10]. This scenario highlights two common needs for researchers and engineers working on the project:

1. understanding the OntoSIDES KG, its content and structure;
2. analyzing and comparing GEs generated from it.

In the following use cases, we considered a subgraph of OntoSIDES including only answers to questions of the *pediatrics* medical specialty. The extracted subgraph contains in total almost 650.000 triples.

3.1 Understanding a Knowledge Graph

We tested *Stunning Doodle* for the visualization and comprehension of the KG in the SIDES scenario. In this case, users are only aware of the high-level concepts

[1] `https://github.com/antoniaettorre/stunning_doodle`

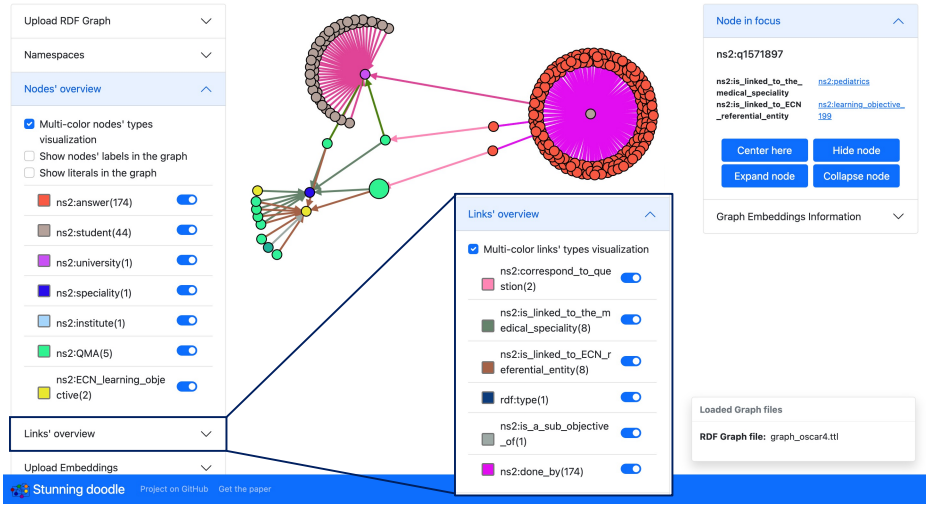

Fig. 5. Screenshot of *Stunning Doodle* showing the basic entities and relations in the OntoSIDES graph.

that are supposedly defined in OntoSIDES and they need to understand what the described entities are, how they are modeled in the KG and how they are linked to each other through predicates. Expert users are normally able to gain this knowledge by running several prototypical SPARQL queries, but the use of *Stunning Doodle* facilitates this task by retrieving and displaying the same information only through a few clicks, and, at the same time, makes the KG accessible also to users with no expertise in SPARQL.

Once the graph file is uploaded, users can choose a node from which to start the graph exploration. This allows users to expand only the nodes that are relevant for them, displaying only the needed information. After expanding a few nodes, users should be able to visualize a small graph containing all the main elements of interest, i.e questions, students, answers, institutes, and their links. Thanks to the "Nodes' overview" menu, users are able to distinguish the main entities in a glance, based on their colors. Considering the visualization (Fig. 5), it is evident that instances of *answers* (orange nodes) are directly linked to *questions* (spring green) and *students* (in gray). Moreover, it is possible to see that questions are associated with *specialties* (navy blue nodes) and *learning_objectives* (yellow). Taking a look at the "Links' overview" menu, predicates can also be easily discriminated in the graph thanks to their colors. Moreover, connections between distant nodes can be quickly identified to be possibly used as property paths in SPARQL queries. For example, to analyze which topics a student worked on, it is sufficient to follow the path going from a gray node (student) to a yellow node (*learning_objectives*), therefore, chaining the properties *done_by* (fuchsia line), *correspond_to_question* (pink), *is_linked_to_ECN_referential_entity* (brown). Additionally, few statistical observations on the number and types of

entities can be done from the visualization. In the example in Fig. 5, the "expanded" student gave 174 answers, the displayed university has 44 students, all the questions are linked to the same medical specialty. Finally, each node can be analyzed in detail, by selecting the node and looking at the menu "Node in focus" which displays the triples associated with that node.

Thanks to its characteristics, *Stunning Doodle* proved to be not only useful but fundamental to gain a first, global understanding of the content and organization of the OntoSIDES KG. Its navigation systems, filtering functionalities, and customization capabilities allow the user to explore the graph step by step, focusing at each moment only on the pieces of information of interest for them. We claim that those features are very helpful for the understanding of Knowledge Graphs in any context.

3.2 Analyzing and Comparing GEs

The GEs analysis provided by *Stunning Doodle* allows to gain insights into the information encoded in the GEs and, therefore, to assess their meaningfulness for their final use. In particular, *Stunning Doodle* can be used for comparing GEs computed through different embedding models with several hyper-parameters settings, with the final goal of identifying and tuning the model generating the most meaningful embeddings.

In the framework of the SIDES project, GEs computed from OntoSIDES have been used as input features for a ML model designed to predict students' performance on medical questions [10]. In this context, (1) interpreting the information captured by GEs, (2) identifying the best model for their computation, and (3) tuning the hyper-parameters to obtain a meaningful representation of the nodes are crucial tasks.

In the following, we show how *Stunning Doodle* has been used to carry out these tasks, by analyzing the GEs generated from the OntoSIDES subgraph described in Section 3.1. We inspect and compare two sets of GEs, both computed using the *node2vec* model [11]. In the first case, the embeddings have been computed considering the graph directed (i.e. it can be traversed only going from subjects of triples to objects), while in the second case embeddings are obtained considering the graph undirected (i.e. links are bidirectional, it is possible to go from objects to subjects). In both cases, we study the embeddings of a student node (`stu81235`) to figure out which nodes are considered to be similar from the embeddings point of view and, therefore, what information embeddings can capture from the KG. After uploading the graph and embeddings' files, we select the node corresponding to `ns2:stu81235` and we choose to visualize its closest nodes in the embedding space.

GEs from a Directed Graph. Fig. 6 shows the result of visualizing the 100 nodes with the shortest euclidean distance (in the embedding space) from the node `ns2:stu81235`. The darkest node is the node `ns2:stu81235` itself since its distance from itself is 0. Immediately, it occurs from the visualization that its closest nodes are not directly linked between each other and, based on the gradient

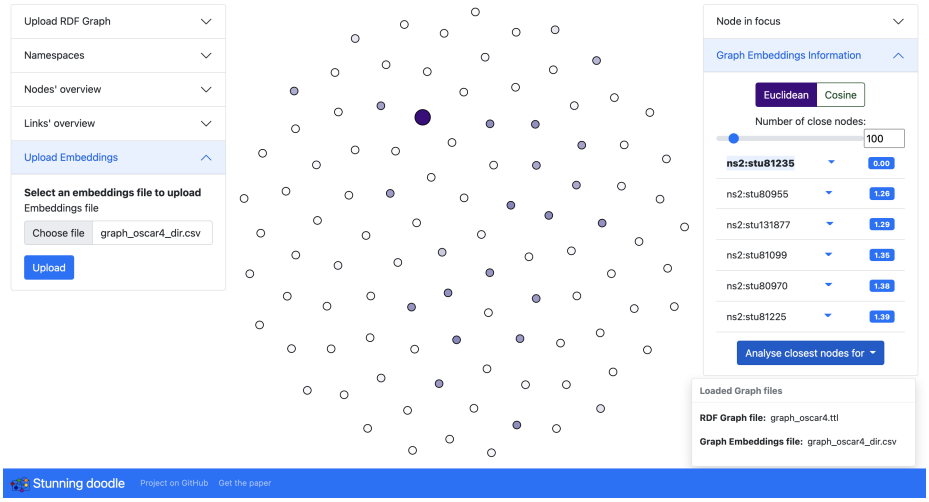

Fig. 6. Closest nodes in the embedding space to the node `ns2:stu81235` with GEs computed from a directed graph.

color, some of them are much closer than others. Looking at the nodes' list in the menu "Graph Embeddings Information", we can see that all the closest nodes are other instances of the student class. Therefore, we can assume that during the embedding process, nodes with the same type are recognized as similar, ending up close in the embedding space.

To investigate why some students are closer to `ns2:stu81235` than others, we can "expand" a few nodes to find connections among them. Fig. 7 shows that the closest nodes (darkest color) are connected to the same university (fuchsia node) as `ns2:stu81235`. This highlights the fact that the link with the university has a high impact during the computation of the embeddings of a student's node. Nevertheless, we can notice that not all the students registered at the same university as `ns2:stu81235` are among its closest nodes (gray nodes in Fig. 7), meaning that there are other factors affecting the similarity between embeddings. Repeating the same analysis for other nodes of type `ns2:student` led to the discovery of similar patterns, i.e. all the analyzed nodes resulted to be similar to other students attending the same university. Thanks to these observations, we can conclude that, in this case, GEs are able to encode the information about the node type and the university for students' nodes.

GEs from an Undirected Graph. Fig. 8 shows the closest nodes to `ns2:stu81235` for the embeddings computed from the undirected graph. The difference with the GEs computed in the previous case is immediately visible. Firstly, by looking at the list of the closest nodes we discover that for undirected embeddings the type of the node does not play an important role in the embedding process. Indeed, the closest nodes are of various types, including `ns2:answer` and

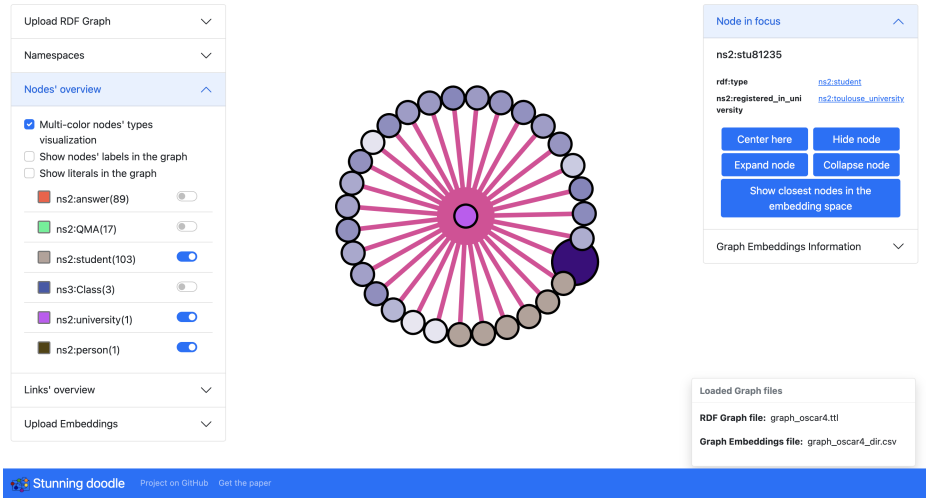

Fig. 7. Links between `ns2:stu81235` and its closest nodes in the embedding space.

`ns2:action_to_answer`. From the visualization, it is obvious that the connections between nodes assume a much more important role, as the closest nodes in the embedding space result to be the ones that are close in the KG as well (at 1-hop or 2-hops distance). On the other side, we notice that the opposite implication does not hold. Indeed, when we expand the node `stu81235` to display all the direct neighbors in the KG as done in Fig. 9, we can see that there are additional nodes which were not displayed before as they are not close in the embedding space (answers in orange). Therefore, we can assume that the distance in the KG, though important, is not the only parameter taken into account during the embedding process. Same observations can be done for other nodes of type `ns2:student`.

Finally, uniquely through the joint visualization of the distances in the embedding space and the KG structure, we can draw interesting conclusions on the meaning of GEs and, thus, identify the best settings to be used for the embedding computation. In our case, we discovered that GEs computed from directed graphs can capture semantic information from the KG, such as the nodes' type and the university associated with each student; while GEs obtained from undirected graphs tend to summarize the graph structure. As a consequence, if our final goal is the prediction of students' performance [10] the information about the attended university is likely relevant and GEs computed from the directed graph are more meaningful.

4 Related Work

Despite the growing attention dedicated to the issues of interpretability and understandability of KGEs, to the best of our knowledge, no work has been

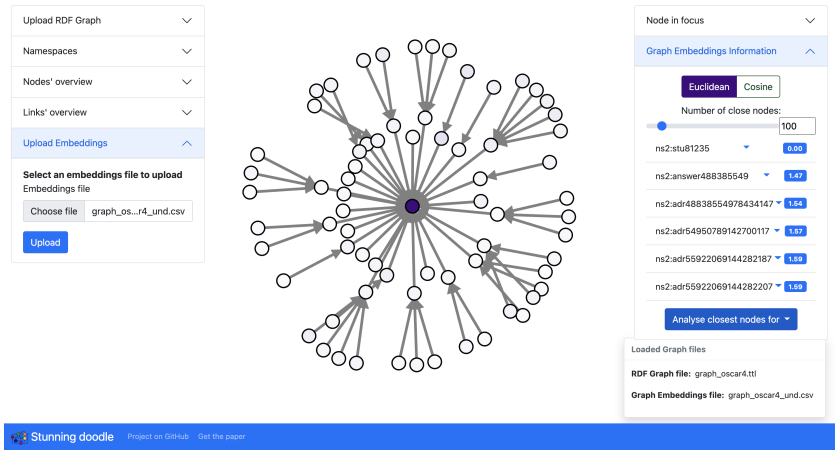

Fig. 8. Closest nodes in the embedding space to the node `ns2:stu81235` with GEs computed from an undirected graph.

done towards the visual analysis and comprehension of such representations. Yet, some solutions have been developed for the similar task of sentences and words embeddings visualization. *TensorFlow Projector*[2] allows users to upload and visualize their embeddings to help the comprehension of the information they summarize. Through this tool, it is possible to visualize the uploaded embeddings in a 3D space by using dimensionality reduction techniques. Moreover, the tool permits, for each element, to list its closest neighbors in the embedding space with the final goal of providing insights on the meaning of the computed embeddings. Although *TensorFlow Projector* is general enough to visualize any kind of embedding, i.e. word, sentence or graph, its usefulness in understanding KGEs is limited by the absence of KG-specific functionalities, such as displaying the graph structure or considering semantic information.

Concerning the visual analysis of KGs, multiple tools have been developed throughout the years. Recent research efforts [1, 6] focused on listing, analyzing, and comparing existing Linked Data (LD) visualization tools able to deal both with KG schema and data. The majority of the available tools rely on a graph-based visualization and present several commonalities with respect to the provided functionalities and input mode. Many of these applications enable to visualize resources accessible through a public SPARQL endpoint [16, 18], while only a few of them allow users to upload local RDF graphs [21]. Some of the tools complement graph visualization with advanced features, such as augmenting graph content with external information [16] and collecting statistics about the KG [20]. Different strategies are used to deal with large-scale KGs, e.g. [18] aggregates nodes in clusters, [2, 21] try to identify and show firstly the most important concepts, while others, such as [15, 5], rely on incremental exploration by the user. The conclusion drawn by [6] highlights the impossibility of identi-

[2] `https://projector.tensorflow.org`

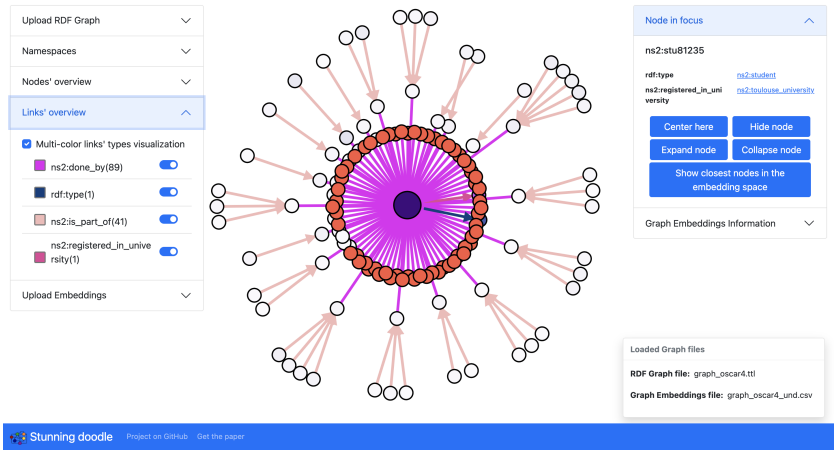

Fig. 9. `ns2:stu81235` expanded to show its neighbors in the KG that are not close in the embedding space.

fying the best tool in the absolute and states that the research in this field is far from conclusion. Moreover, most of the existing applications are developed as proofs of concept or research tools, therefore they are often conceived for a very specific task and dataset and they can hardly be generalized, or they are rather cumbersome to set up. These limitations make them not suitable for widespread use by non-expert users.

Inspired by the GEs visualization implemented by Tensorflow, *Stunning Doodle* builds upon the functionalities offered by recent KGs visualization tools and extends them to enable a simple and straightforward visual analysis of the KGEs.

5 Conclusion and Future Work

Stunning Doodle is a first step to fill the gap in the field of visual analysis of KGEs. This visualization tool enables to build a link between the content and structure of any KG and its corresponding embeddings. We implemented a set of functionalities to facilitate the exploration and understanding of any KG and to analyze KGEs, connecting the two, and making sense of the information captured by the KGEs. We used *Stunning Doodle* to address two use cases requiring the study of a real-world KG and its embeddings. It has proven its usefulness in gathering meaningful insights for a more informed exploitation of KGEs.

As future work, we plan to implement new functionalities particularly useful for more expert users, such as the visualization of the result of SPARQL queries and the direct access to well-known SPARQL endpoints. Moreover, we aim to provide a deeper analysis of the uploaded GEs including advanced statistics on the closest nodes and additional similarity metrics. We also want to optimize the pre-processing pipeline to be able to display and analyze larger KGs. Our hope is that *Stunning Doodle* could build a large community of users and keep improving and growing throughout the years to satisfy their needs.

References

1. Antoniazzi, F., Viola, F.: RDF Graph Visualization Tools: a Survey (11 2018). https://doi.org/10.23919/FRUCT.2018.8588069
2. Asprino, L., Colonna, C., Mongiovì, M., Porena, M., Presutti, V.: Pattern-based visualization of knowledge graphs. arXiv preprint arXiv:2106.12857 (2021)
3. Bikakis, N., Liagouris, J., Kromida, M., Papastefanatos, G., Sellis, T.: Towards scalable visual exploration of very large RDF graphs. In: European Semantic Web Conference. pp. 9–13. Springer (2015)
4. Bikakis, N., Liagouris, J., Krommyda, M., Papastefanatos, G., Sellis, T.: GraphVizdb: A scalable platform for interactive large graph visualization. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE). pp. 1342–1345. IEEE (2016)
5. Camarda, D.V., Mazzini, S., Antonuccio, A.: Lodlive, exploring the web of data. In: Proceedings of the 8th International Conference on Semantic Systems. pp. 197–200 (2012)
6. Desimoni, F., Po, L.: Empirical evaluation of linked data visualization tools. Future Generation Computer Systems **112**, 258–282 (2020)
7. Ernst, P., Meng, C., Siu, A., Weikum, G.: Knowlife: A knowledge graph for health and life sciences. In: 2014 IEEE 30th International Conference on Data Engineering. pp. 1254–1257 (2014)
8. Ettorre, A.: antoniaettorre/stunning_doodle: First version (Dec 2021). https://doi.org/10.5281/zenodo.5769192, `https://doi.org/10.5281/zenodo.5769192`
9. Ettorre, A., Bobasheva, A., Faron, C., Michel, F.: A systematic approach to identify the information captured by knowledge graph embeddings. In: IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT) (2021)
10. Ettorre, A., Rodríguez, O.R., Faron, C., Michel, F., Gandon, F.: A Knowledge Graph Enhanced Learner Model to Predict Outcomes to Questions in the Medical Field. In: International Conference on Knowledge Engineering and Knowledge Management. pp. 237–251. Springer (2020)
11. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 855–864 (2016)
12. Kazemi, S.M., Poole, D.: Simple embedding for link prediction in knowledge graphs. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. pp. 4289–4300 (2018)
13. Liu, J., Lu, Z., Du, W.: Combining enterprise knowledge graph and news sentiment analysis for stock price prediction. In: Proceedings of the 52nd Hawaii International Conference on System Sciences (2019)
14. Mahdisoltani, F., Biega, J., Suchanek, F.: Yago3: A knowledge base from multilingual wikipedias. In: 7th biennial conference on innovative data systems research. CIDR Conference (2014)
15. Micsik, A., Turbucz, S., Györök, A.: Lodmilla: a linked data browser for all (2014)
16. Nuzzolese, A.G., Presutti, V., Gangemi, A., Peroni, S., Ciancarini, P.: Aemoo: Linked data exploration based on knowledge patterns. Semantic Web **8**(1), 87–112 (2017)
17. Palombi, O., Jouanot, F., Nziengam, N., Omidvar-Tehrani, B., Rousset, M.C., Sanchez, A.: OntoSIDES: Ontology-based student progress monitoring on the

national evaluation system of French Medical Schools. Artificial intelligence in medicine **96**, 59–67 (2019)
18. Po, L., Malvezzi, D.: High-level visualization over big linked data. In: International Semantic Web Conference (P&D/Industry/BlueSky) (2018)
19. Rotmensch, M., Halpern, Y., Tlimat, A., Horng, S., Sontag, D.: Learning a health knowledge graph from electronic medical records. Scientific Reports **7** (12 2017)
20. Santana-Pérez, I.: Graphless: Using statistical analysis and heuristics for visualizing large datasets. VOILA@ ISWC **2187**, 1–12 (2018)
21. Troullinou, G., Kondylakis, H., Stefanidis, K., Plexousakis, D.: RDFDigest+: A Summary-driven System for KBs Exploration. In: International Semantic Web Conference (P&D/Industry/BlueSky) (2018)
22. Ying, R., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnnexplainer: Generating explanations for graph neural networks. Advances in neural information processing systems **32**, 9240 (2019)