Knowledge Graph Visualization

Piyush Chawla, Skylar Wurster, Patrick Capps

April 21, 2019

Abstract

Word embeddings are a hot topic in natural language processing research. The last decade has witnessed many different word-embedding models and their linguistic capabilities (e.g. the infamous King-Queen, and Man-Woman relation). Another knowledge-representation technique is Knowledge-Graph completion. Although knowledge graph completion algorithms have done very well in the last decade, very little time has been spent on analyzing the geo-spatial properties of these embeddings. In this work, we aim to analyze knowledge graph embeddings for the FB15k dataset, trained on transE model. For this we use two broad techniques, visualizing the trained embeddings, and using a pre-trained word-embeddings model. We finally propose a new tool that visualizes embeddings in 2D, 3D and VR.

1 Introduction

In the last decade knowledge graphs have gained importance in the semantic web, natural language processing and machine learning community. This sudden rise in popularity of this 'data structure' can be attributed to the Google researchers that linked Google's Knowledge Graph to the Google search module. Other important contributions to this area were larges-scale knowledge graphs like Freebase (Google) and DBpedia (Lehmann et al.) that popularized this idea. DBpedia uses their wikipedia 'extraction-framework' to regularly update their Knowledge Graph as new versions of Wikipedia are released.

A knowledge graph consists of millions and billions of triples of entities. Each triple is divided into a 'subject', 'object' and 'predicate'. For instance 'Donald Trump', 'United States of America' and 'president' is a triple that points to the fact the 'Donald Trump is the current president of United States of America'. Since each triple essentially represents a fact about the world, knowledge graphs are heavily used in tasks like question-answering, named entity recognition and many other branches of natural language processing.

An interesting problem statement involving knowledge graphs is whether we can predict new facts about the world given the existing facts, which translates into coming up with a probabilistic model that can give the probability of occurrence of a new triple given the knowledge graph. For example if the knowledge graph knows that 'Delhi is the capital of India,' which means that there exists a triple 'Delhi','capital', 'India' in the knowledge base, then the model can predict that 'Delhi is a city'. In the last decade many interesting algorithms have been proposed that aim at solving this problem. One such algorithm is transE which is a vector translation based algorithm that encodes all the subjects, predicates and objects in the same vector space.

Though the knowledge graph embeddings have performed very well in predicting new information, little has been done to visually analyze the actual embeddings produced by these models. TransE for instance has been based on a novel idea of vector translations but the community lacks effective tools that can verify this hypothesis and compare these embeddings to other knowledge graph representations.

In this work we aim to propose a new tool for visualizing knowledge graph embeddings trained on FB15k dataset, a widely used baseline dataset in the Knowledge Graph community.

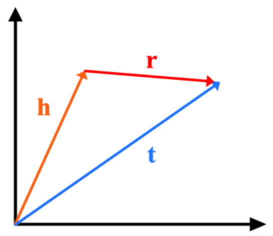


Figure 1: Translation of subject vector (h) to object vector (t) when relation vector (r) acts on it.

2 transE

One of the well-known model for knowledge graph completion is transE (Bordes et al.) [1]. This model was published in NIPS 2013 and gained popularity due to its simple design. The algorithm maps the entities (subjects and objects) and relations (predicates) in a knowledge graph triple to N-dimensional vector space. The idea is unique because it considers the embedding vectors as translations in the vector space. The subject vector is translated to the object vector, and this translation is provided by the relation vector. Figure 1 shows the schematic diagram of this idea.

The loss function is just the mathematical representation of this translation idea. The equation below gives the *Loss*.

$$Loss = |subject + predicate - object|_{L1/L2} \tag{1}$$

However, since there is no negative data, the model would learn within a couple of epoch and would always predict any fact as true. To overcome this problem, the authors proposed a negative sampling based method. Here, negative samples are corrupted triples generated by replacing subject/object/or both by random entities. The equation below gives the final loss.

$$FinalLoss = Max(0, \lambda + Loss_{pos} - Loss_{neg}) \tag{2}$$

Here, $Loss_{pos}$ is the loss, in equation (1), computed on the positive triples and $Loss_{neg}$ is loss computed on negative triples. Once we have the final loss, we learn the embeddings (weights) by training the model using gradient descent.

3 Skip-Gram

Word embedding models have been around since 1986. This class of model is known as 'Language Modelling' and has been progressing in the last few decades. The word2vec model a.k.a skip-gram [4] was published in NIPS 2013. The model uses a simple neural network architecture to map words in the vocabulary to N-dimensional space.

The algorithm works by moving a word context window over the training text. Which in simple term means that for a word under consideration, the model looks at the words in its immediate neighbourhood (both before it and after it). The algorithm then tries to predict the neighbourhood words given the input word. Figure 2 describes this idea in the form a simple architecture.

The input to model are one-hot encodings of the words, which means that sparse vector of size V (vocabulary) having only one 1 and rest values are 0s. The output are K (Window size) soft-max vectors one for each neighbourhood word. The figure shows a projection layer, this is where the one hot vector gets reduced to a lower dimensional space N¡¡V. The model learns weights for this projection matrix. This project matrix is the word-embedding matrix, containing encodings for each word in the model's vocabulary.

The original paper [4] proposed two models, CBOW and SkipGram but SkipGram gave better results at analogy prediction task is thus used by NLP and other branches of AI. The reason that this model is widely used by the NLP community is because the embeddings generated contain a

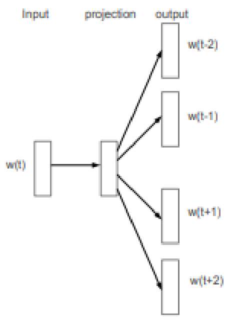


Figure 2: The skip-gram model. The neighbourhood of input (w(t)) is predicted using a neural network architecture.

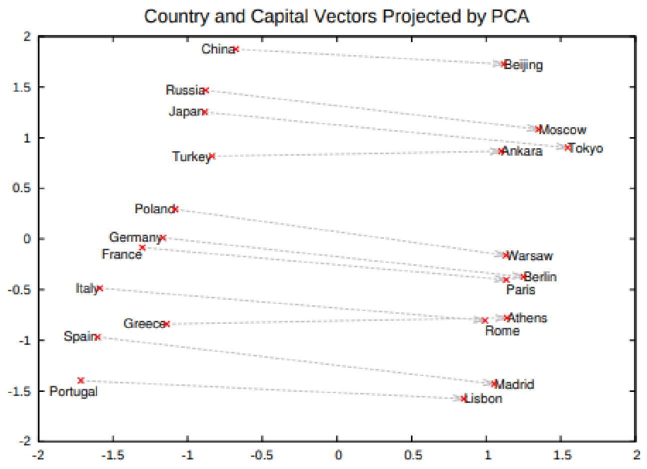


Figure 3: An example of analogy analysis using word embeddings. Country-capital analogy is visualized after running PCA.

lot of semantic information about the dataset encoded in them. One of most analyzed properties of word embeddings is the analogy relationship. Figure 3 shows an example of such relationship, the 'country-capital' analogy.

4 Related Work

Other applications exist to address this problem space. One example is the Embedding Projector [6], a WebGL tool to interactively view embedding projections in 2D and 3D. The Embedding Projector aims at providing needed interactivity, such as zooming, rotating, and panning, to visualizations that are generally static. This tool is useful, however we hope that with a VR application we can add more interactivity and make it feel natural, so that researchers can view embeddings from new perspectives.

Word Embedding Visual Explorer [7] is another web-based word embedding visualization tool. For this tool the authors' goal was to create projected views that fixed the distortion effects of standard PCA and t-SNE, as well as to preserve analogy pair relationships in the data. Although our tool uses PCA and therefore has the same drawbacks Liu et. al discuss in their work, we felt it important to include subject and object relationships from the knowledge graph in the visualization.

Heimerl and Gleicher's work [8] focuses on creating a visualization to address common tasks that

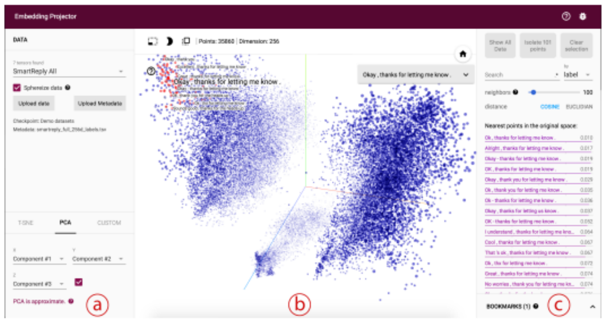


Figure 4: A PCA projection of a corpus of 35k frequently used phrases in emails in the Embedding Projector

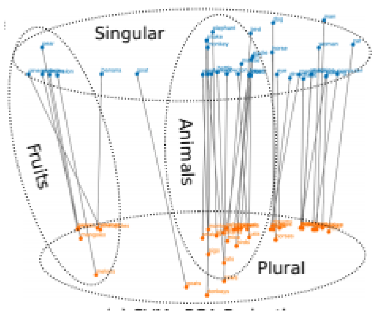


Figure 5: A modified PCA projection showing two types of relationships with the Word Embedding Visual Explorer

word embedding researchers wish to perform. Many of their visualizations focus on word co-occurrence. In our tool, we aim to provide options for several of the tasks in their paper, such as exploring local neighborhoods, however a 3D VR visualization will of course differ in many ways from a 2D visualization

5 Dataset

The dataset we chose for our knowledge graph is FB15k, a well-known bench-marking dataset used for knowledge graph completion algorithms. This dataset was first published in the transE paper [1], and is a subset of Freebase Knowledge graph [5]. FB15k was created by selecting fifteen thousand most frequent entities (subjects and objects) and their corresponding triples from the entire Freebase KG.

For the word embedding analysis we needed the entity and relation names to apply the pre-trained embedding model on. The relations in FB15k are present in natural language form but the entities are in MIDs. The first challenge was to extract the corresponding English form for these MIDs.

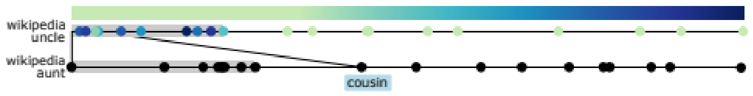


Figure 6: Comparing the differences in neighborhoods between uncle and aunt with Heimerl and Gleicher's tool

6 Design Method

Given that both models map strings to a 300 dimensional vector space, Principle Component Analysis (PCA) is used for dimensionality reduction. For this project, PCA is used on the entire dataset, and not recalculated on subset results from applied filters as [7] do. A 3-dimension PCA reduction is used to create a resulting map from strings to 3-space. We identify 4 tasks for analyzing transE and Skip-Gram on FreeBase15k.

Filtering by predicate
FreeBase15k connects subjects and objects via predicates in the form of a Subject-Object-Predicate (SOP) triple. There are over 700 unique predicates in the dataset, and over 550,000 SOP triples. We allow the user to select a single predicate from the list and show all subjects and objects connected by that predicate. The user can chose to color the items by their closest topic as seen on the left of the UI, or by subject (blue) and object (red). The user may also decide if they'd like to see the connections in the embeddings as well.

Compare local neighborhoods
To analyze the differences between the two embeddings, the local neighborhoods within the embedding reveal information about what the algorithm at hand believes to be similar. This can help with development and debugging of a certain algorithm. As shown in [8], some embeddings that aren't deterministic may have poor results when compared to other embeddings, and this may become clear through visualization. To select a neighborhood, the user first ensures that the "Selection" option is set to "From sphere only". Then, they may use their controller to create a sphere of any size by gripping at a location, and pulling the controller toward or away from the grip point. When this is released, the embedding elements within that sphere will be the only elements selected. This may be used to select exactly one element or a large subset. In the embedding that the user selected from, the result will be a local neighborhood. The other embedding will also update to show that same subset of elements, and the user may analyze the grouping of the result.

Compare connected subjects and objects
Similar to the comparison of local neighborhoods, a user may also learn information about the algorithm and dataset by seeing the topology of the resulting network when an element(s) and all of its connected subjects and/or objects are selected. To select connected subjects and objects, the user must first set the "Selection" option to "From sphere + connections". Then, in similar fashion as comparing local neighborhoods, the user creates a sphere that will select any element inside of it using their grip. It is suggested to only select 1-5 elements at a time with this, as there may be a very large resulting subset of connections that may cause disorienting lag. Then, both embeddings will update to show the selection and all of its connections to other elements. The user may then compare the topologies of the algorithms' embeddings to aide in development and debugging through examining clusters, element labels, and outliers.

Querying individual elements
With VR, we have new interaction techniques that don't exactly rely on 2D pixel space. So, we add functionality that allows the user to hover their hand near the elements, and the closest elements will show their corresponding text above their cube. This allows for the user to see what types of elements are in that area (cluster), and allows for comparison of multiple texts at one time. Traditional mouse selection may only support one text element being analyzed at a time, but with 3D and head tracking, the user can move around to read text.

We aimed for a sleek and intuitive design for the space the user is in. The user interface (UI) directly in front of them shows the coloring legend, options for filtering, and other interaction options. Above the UI are the controls. We put this front and center so they can change these easily. The ambient space is neutral gray so that users can focus on the elements in the embedding they're analyzing.

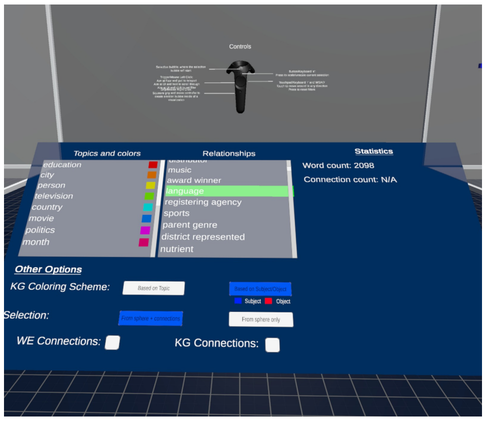


Figure 7: UI menu in the VR space

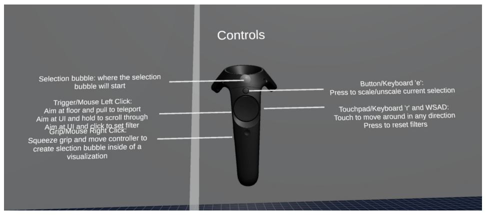

Controls
Selection bubble: where the selection bubble will start
Trigger/Mouse Left Click: Aim at floor and pull to teleport Aim at UI and hold to scroll through Aim at UI and click to set filter
Grip/Mouse Right Click: Squeeze grip and move controller to create slection bubble inside of a visualization
Button/Keyboard 'e': Press to scale/unscale current selection
Touchpad/Keyboard 'r' and WSAD: Touch to move around in any direction Press to reset filters

Figure 8: Labeling for the controls

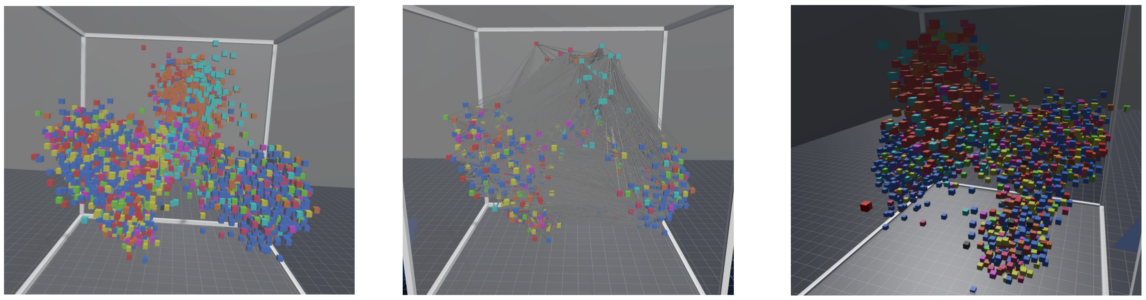
The embeddings are housed within transparent cubes on either side of the UI. We elect to not show axis labels to save space and prevent users from focusing on something which has little meaning. For navigation, we offer both teleportation and touch-pad movement. Though touch-pad movement can be jarring, we find it useful as users can keep their eyes focused on certain embedding points while moving.

7 Implementation method(s)

Our tool was developed using Unity 3D and C# for the main application, while the back-end embedding processing was done using Python. In order to make the visualization VR compatible, the application uses the Virtual Reality Toolkit by Sysdia Solutions on the Unity Asset Store. The tool was tested using SteamVR and an HTC Vive.

8 Results and analysis of results

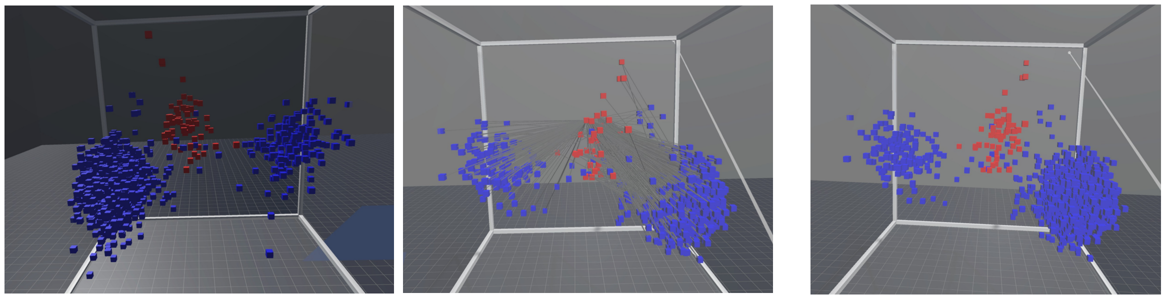
For the first prototype demonstration, we did not yet have access to a VR headset for testing, so all debugging was done via mouse and keyboard. Since this prototype, we've added: a full UI with labels for topics and colors, selection by connection, optimizations, loading bars, teleportation, movement via touchpad, and many bug fixes.



(a) Front view, connections on (b) Front view, connections on (c) Rear view, connections off

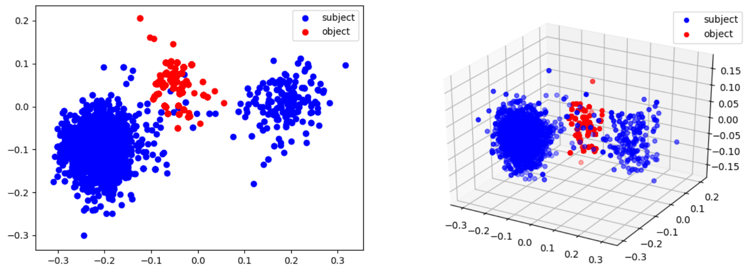
Figure 9: Knowledge graph visualization of "USA" and connected subjects and objects viewed from the front

We see from Figure 9 that 3D + VR allows a new kind of interaction technique that allows a deeper understanding of the embedding. It is more clear to see connections especially in large subsets, such as that shown in Figure 9. We can see the clear topology and clusters in VR, and the flow that connections are tending to make.



(a) Front view, connections off (b) Front view, connections on (c) Rear view, connections off

Figure 10: Knowledge graph visualization of the predicate ”Language” and what subjects and objects it connects



(a) 2D projection of Language 'triples'. (b) 3D projection of Language 'triples'.

Figure 11: Visualizing the 2D and 3D projection of KG embeddings for the 'Language' predicate.

9 Discussion of results

This work was presented to a domain expert in the fields of design, 3D modeling, and virtual reality. Feedback was overall positive, citing the clear instructions and good contrast between the background and foreground keeping the eyes focused on the visualization, instructions, and user interface. The use of hover effects to show when a user interface item could be selected also was an important feature referenced. Some improvements suggested were related to the visualizations themselves. It was recommended to remove the glass panes around the visualizations, as this can lead to a ”museum effect” in which the user gazes from afar and doesn't realize the visualization is intractable. While we find this useful for displaying the minimum and maximum x, y, and z values, we can see how this may be the case.

Another piece of input given from this expert was the movement functionality. The touchpad allows users to navigate around, but this movement is very motion sickness inducing to many virtual reality users. He recommended we remove that completely, or allow it to be toggled, so that users don't accidentally do this. We agree, and think this should be given as a warning to users beforehand.

As a last piece of input, it was suggested that more feedback is given to the user when hovering over intractable objects on the user interface. A brighter glow or haptic feedback will invite the user to interact and know when they have hovered over something intractable.

10 Conclusions and future work

In this work we visualize knowledge graph for the FB15k dataset. We used two embedding based methods, KG embeddings and word embeddings to map the entities and relations into the vector space. The KG embeddings were trained on the transE model and word embeddings used a pre-trained skip-gram model. Our analysis shows that VR surpasses 3D and 2D visualization both in terms of time and accuracy, especially while analyzing specific relations. In our future work we plan to extend this analysis to analogy performance comparison on word-embedding models. The word2vec model lacks contextualization, due to this embeddings for multi-word entities like 'UCLA basket ball team' are computed using heuristics like taking average of the words. This leads to sub-optimal embedding mappings. We plan to use contextualized embedding models like ELMo [3] and BERT [2] in the future work.
Visualizations may be updated as well. Ribbon connections to group the connections could be made using a shader to leverage the efficiency and allow the user to see clusters of connections. Allowing users to enter topics in realtime and see the resulting coloring would be useful for analysis and debugging. Adding a way to select multiple elements that aren't spacially near each other (as the current selection tool requires) would also be beneficial.Realtime PCA computation may help in some circumstances. Lastly, allowing users to select their own embeddings to add into the visualization, perhaps in a circle around them, would be useful for comparing even more embeddings in the same space and not necessarily at the same time.

References

[1] Bordes et. al, Translating Embeddings for Modeling Multi-relational Data
`https://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf`

[2] Devlin et. al, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
`https://arxiv.org/abs/1810.04805`

[3] Peters et. al, Deep contextualized word representations
`https://arxiv.org/abs/1802.05365`

[4] Mikolov et. al, Distributed Representations of Words and Phrases and their Compositionality
`https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf`

[5] Bollacker et. al, Freebase: a collaboratively created graph database for structuring human knowledge
`https://dl.acm.org/citation.cfm?id=1376746`

[6] Smilkov et. al, Embedding Projector: Interactive Visualization and Interpretation of Embeddings
`https://arxiv.org/pdf/1611.05469.pdf`

[7] Liu et. al, Visual Exploration of Semantic Relationships in Neural Word Embeddings `http://www.sci.utah.edu/~beiwang/publications/Word_Embeddings_BeiWang_2017.pdf`

[8] Heimerl and Gleicher, Interactive Analysis of Word Vector Embeddings `https://graphics.cs.wisc.edu/Papers/2018/HG18/embeddings_preprint.pdf`