Visualizing High-dimensional Ball-Embeddings while Keeping Topological Relations

First Author

July 29, 2020

Abstract

Region-based embeddings have been proven to be useful in knowledge graph reasoning. Existing visualization tool cannot preserve topological relations among high dimensional regions. We present a tool for visualizing high-dimensional balls that keeps topological relations after their dimensions are reduced to 2-dimensions. The first version of this tool provides four functions as follows: (1) Simple diagrammatic reasoning for syllogism; (2) diagrammatic reasoning with back-ground knowledge; (3) visualizing the construction process of ball-embeddings; (4) providing a web-service to impose a taxonomy structure onto its vector embeddings with zero-energy loss. A demonstration video of this tool is available at ....

1 Introduction

Bing able to learn from data, and robust to noisy inputs, Deep-Learning has been successful in a variety of AI tasks that have frustrated classic symbolic approaches for decades, such as object classification, machine translation, voice recognition, question-answering [**?**]. However, Deep-Learning Systems lack of explainability, can be fooled [**?**, **?**, **?**], and normally need much more learning data than human does [**?**]. This introduces potential dangers into safety critical applications, such as autonomous driving. Introducing innate structures into Deep-Learning system has been advocated, and listed as one of the AI research topics in the Townhall meeting at AAAI-19, so that Deep-Learning research shall solve logical reasoning tasks (System 2 of mind) [**?**, **?**]. As logical reasoning, such as Syllogism, is better represented by inclusion relations among regions, instead of translation among vectors [**?**, **?**], recent researches have been attempting to promote vectors into regions to improve performances of logical query, triple classification, link prediction of Knowledge Graph [**?**, **?**, **?**, **?**, **?**].

The popular tool t-NSE [**?**] for visualizing vector embeddings cannot be directly applied for visualizing ball embeddings, for the dimensional reduction process of t-NSE does not guarantee the topological relations among balls. In this paper, we demonstrate an open source system that is able to visualize ball

embeddings while keeping their topological relations. The main contributions of this system are as follows: (1) it has an vivid interactive user interface that can be used for diagrammatic reasoning among taxonomy; (2) it provides an effective and friendly approach for debugging the geometric construction process of ball embeddings; (3) it provides a batch service that accepts a large scale input for construct ball embeddings.

2 N-Balls embeddings

Hierarchies or in other words trees can be expressed by ball embeddings. The largest N dimensional sphere corresponds to the root node of the tree. The direct children of the root node are represented by smaller spheres that are contained within the sphere of the root node. Hierarchies are an important part of natural language and signifies a type of relationship with another word. For example sparrow duck or Eagle are all of the type bird. In linguistics this is called a hypernym and hyponym relationship where the type of, in our example the bird, would be the hypernym and the concrete realizations of that type, in our example sparrow or a duck, would be some hyponyms of bird. It is self evident that having this kind of hierarchical information can be useful when creating a rule-based natural language processing tool. Unlike stochastical or learning based methods hierarchical structures do not come with an error rate or in other words we can always determine if a note is a child (hyponym) of a word that we are interested in or not. For example if we were to write a tool that needs to identifies cities within a certain region then the results most certainly have to be direct or indirect hypnonym of the word city.

While rule-based natural language processing have been studied and applied for many decades now recently machine learning specifically deep neural networks have produced stunning results that seemed almost impossible with the traditional rule-based methods. However to actually use words or sentences in a neural network we first have to convert the input into to a vector that we can train our model on. This was first explored in the word2vec paper. The goal is to create a dense vector representation for each word. They start off by creating a sparse representation also known as one hot encoded. For this they determine all words used in their text corpus and create a vocabulary from that. Then every word can be represented by a vector of the length of the vocabulary that only contains a single entry 1 and all of her entries are 0. This allows us though in a very inefficient way to feed any word or even a sentence, a concatenation of these vectors, into a neural network. Further they assume that similar words have a similar context. For example in "Can you *blank* me the train station." we can fill this gap with lead, guide,show and all of them express a similar meaning. We can then train a neural network to represent every word with a limited dimension of our choosing. Common vector dimenions are 50, 100,300,500. This works remarkably well and allows us to explore relationships beyond simple hierarchies. In the original paper for example they could use the word vectors of Paris-France+ Italy=Rome. This approach has been extended

multiple times for example glove. As with all learning based methods there is no guarantee that certain relationships that we observe in our language are actually projected into the word vector space.

N-Ball and beddings combines both approaches to the reliability of a hierarchical based system in combination with the power and flexibility that is encoded into the word vectors. This is achieved by extending the word vectors and conceptually adding a sphere to the end of each of them. In the next step we need to ensure that the word embedding actually adhered to the hierarchical structure that we want to embed into them but at the same time we wish to preserve the vector of it has been assigned by the word embedding of our choice. This is done by adding more dimensions to these word vectors by only using homothetic transformations.

3 The Architecture

In order to provide useful visual feedback for the construction process of the ball embeddings the user needs to provide a history of the construction of their ball embeddings. In order to make the system more accessible and allow for quick demonstrations purposes we allow the user to alternatively provide a set of words along with their taxonomy. Using the approach of [] we then compute a set of high quality word embedding along with their construction history.

To make the system accessible from almost any modern devise we chose to expose the user interface through a website. The user request is accepted by a flask web-server and placed in compute queue to allow for concurrent access and real-time feedback for the user even if the server is busy.

Finally given the ball embedding history we determine for every step if any two N-balls intersect, contain each other or are separate. This allows us to create a visual feedback reminiscent of a debugger of the N-balls construction along with a input taxonomy as a tree. Not only can the user see when his approach violates the taxonomy but also how and with which participant.

3.1 Flask

In order to provide a easily accessible demonstration website or visualization and interactive debugging tool is available through a website. As the original N-Ball embedding is implemented in Python, flask was chosen as a web framework. Flask is a Python based micro framework and as such does not require any additional libraries and is readily available through the python package manager. As is convention with a flask application we provide HTML templates in the template folder. These templates are just the basic HTML skeleton which defines where and in which order a different graphs, input fields and text fields will be inserted on the website. The basic formatting is done through the CSS bootstrap framework. We further extend on this with our own CSS file which can be found under templates/static/css along with the JavaScript code under templates/static/js.

3.2 Redis

For every input the system needs to generate the corresponding ball embeddings, save each step in the construction process, reduce the dimension of the embedding down to two dimensions for visualization and where possible resolve overlapping that results from the dimensionality reduction. Depending on the size of the input tree this process can take anywhere from a few seconds to a few minutes. With a naïve implementation the Web server would simply do all of these tasks before sending back the response to the user. This would not only leave she was awaiting with no feedback but even more importantly the server would not be reachable at all during that timeframe for other users. While works well for single user it is obviously unacceptable for a publicly available website. We need to make sure that a large will not block the Web server for everybody else for minutes possibly even hours. Therefore we need to decouple the serving of the website from the actual computation of the ball embeddings. This is done by using a task queue. We chose to use a commonly used task queue named redis. This allows us to create a task queue so when the user requests the computation of a ball embedding we added to the task you and immediately returned to the user to tell him that we received his request and that he has been placed into the task queue. Once the task is complete the results can be sent to the users browser through AJAX without the need to refresh or redirect the website. This allows for seamless display of the results. While the website will return immediately and give the user adequate feedback this still allows a single user to effectively block the queue. As he places a large request once the Web server starts computing this task it will not work on tasks that have been requested later. In order to keep the websites response time for small requests low we introduced a second high priority queue that is reserved for small to moderate input sizes to ensure that users they want to experiment with the service will not be blocked by a single large request.

```
r = redis.Redis()
q_high = Queue("high", connection=r)
q_low  = Queue("low", connection=r)
```

4 Services

4.1 words meaning and etimology

4.2 Simple Diagrammatic Reasoning

Ball embeddings allow for strict reasoning and logical queries. This can be demonstrated with our tool. The user can input hypernym relationships in plain English such as "socrates is human, human is animal" we display that relationship as seen in figure X. Similarity system also handles exclusion such as "soccer is not human,human is mortal" as seen in figure X. Once these relations have been enforced on the sufficiently large corpus, ideally for all words present

in the underlying word embedding, this allows for strict logic varies which are otherwise impossible in neural networks.

4.3 Diagrammatic Reasoning with Background Knowledge

- user input: Soccer is not human, human is animal
- user query: what is the relation between Soccer and animal
- system replaces 'Soccer is not human' with 'Soccer is entity', system adds 'animal is entity' by searching background knowledge

```
>>> from nltk.corpus import wordnet as wn
>>> soccer = wn.synsets('soccer')
>>> soccer
[Synset('soccer.n.01')]
>>> soccer = wn.synsets('soccer')[0]
>>> soccer
Synset('soccer.n.01')
>>> human=wn.synsets('human')[0]
>>> human
Synset('homo.n.02')
>>> soccer.lowest_common_hypernyms(human)
[Synset('entity.n.01')]
>> % check 'animal' and 'entity'
>> ..
```

- system draw: (1) Soccer ball outside animal ball, (2) human ball inside animal ball; (3) they are inside entity ball
- system merges (1), (2), and (3)
- system conclude: Soccer is not animal.

4.4 Visual Debugging

Enforcing a relational structure onto word embedding is a difficult task. Similar to a code-debugger we provide a visual feedback for every construction step which; (1) allows the viewer to retrace and understand the construction process and as such serves as a valuable teaching tool; (2) enables developers to optimize and correct their ball embedding construction.

Since we only consider hierarchical structures in this paper we can guarantee that any valid state can be visualized on to a two dimensional canvas. In order to achieve this we first construct a set of nonoverlapping circles that perfectly represents the hierarchy that is to be encoded into the ball embeddings. Then within each construction step we either display the previously computed perfect representation or enforce any overlapping's that have been recorded in the ball embedding construction history.

4.5 Batch Service

- user provide her/his name and contact email.
- user input: a tree structure, vector embeddings of tree nodes
- System will construct ball embeddings at backend, and send the user the link fo the final ball embeddings

5 Conclusion and Outlooks

link to the video