Does a dog desire cake? - Expanding Knowledge Base Assertions Through Deep Relationship Discovery

Pedro Colon-Hernandez
Media Lab
Massachusetts Institute of Technology
Cambridge, MA, O2139
pe25171@mit.edu

Henry Lieberman
CSAIL
Massachusetts Institute of Technology
Cambridge, MA, O2139
lieber@mit.edu

Catherine Havasi
Media Lab
Massachusetts Institute of Technology
Cambridge, MA, O2139
havasi@media.mit.edu

Abstract

Relationship discovery between two entities is a problem that has to be addressed when constructing a Knowledge Base (KB). A solution to this problem is important because the KB built from the discovered relations can play a key role in downstream tasks, such as analogical reasoning. An example of this kind of reasoning is whether a dog desires cake: a dog is an animal, cake is food, animals desire food therefore a dog desires cake. We constructed a system that that is trained on a commonsense KB and whose inputs are pairs of concepts and its outputs are the strength of commonsense assertions between the concepts. Our approach is unique because it can handle out of vocabulary entities and can generalize commonsense to out of knowledge concepts. We utilize the system to be able to infer the answer for out of knowledge assertions such as the aforementioned whether a dog desires cake.

1 Introduction

If we were to ask whether a dog eats cake or not, how would we be able to get an answer? Using human intuition we could try and find the answer using analogical reasoning. Analogical reasoning can be seen as a way of reframing a problem to a more familiar context where we can reach an answer or a solution and apply it back to the original framing(1). Taking the example of the dog, we could say that a dog is an animal and that a cake is food. In this more abstract frame, we can easily reach the conclusion that animals desire food. Now that we have that conclusion, we bring the information back to our original dog and cake frame and can say to a certain degree a dog must desire cake. To be able to perform this kind of reasoning, you need to have a way to know the relationships between entities that you are trying to reason about and the relationships of the neighbors of these entities.

Therefore we set out to create a system that can discover relationships between pairs of entities. If we can discover relationships between concepts then we could find similar concepts and could use them later on in an analogical reasoning system. Our system is a Multi-Task Learning system trained on the assertions found in ConceptNet(2), a commonsense knowledge. The inputs to our system are pairs of concepts in the form of retrofitted word embeddings and the outputs are the strength of the different relations between the input concepts. To form the inputs, we first learn a FastText(3) word vector representation of the entities. This representation can is through an off the shelf pre-trained

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

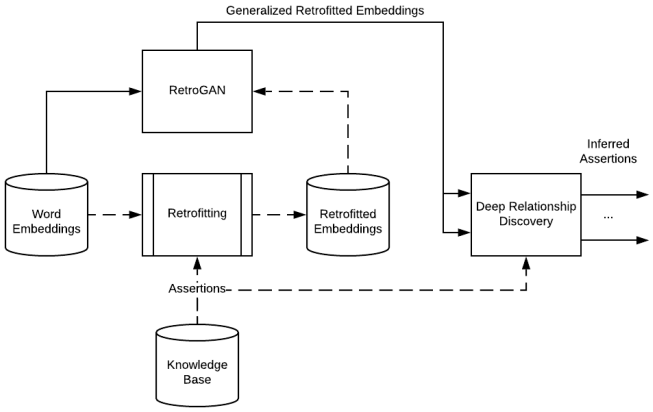

Figure 1: Complete System Architecture. The dashed lines represent procedures that are only necessary when the system is being trained.

model. We utilize FastText for its ability to incorporate sub-word information with the purpose of being able to generalize to out of vocabulary embeddings. Once we have this, we then proceed to retrofit the embeddings with the information in a KB. We retrofit the embeddings with the idea that they will incorporate some of the knowledge in the KB such that when they are used in a downstream analogical reasoning task then they will perform better. Now, we incorporate one of the innovative parts of our work which is a system that implicitly expands the knowledge in the vectors by learning retrofitting mappings. We do this because the retrofitting process only modifies embeddings for entities that are in the knowledge base.

To accomplish this, one of our contributions is a system that we developed called RetroGAN. RetroGAN is a CycleGAN(4) based system that learns the mapping from word embeddings to retrofitted word embeddings. By learning this mapping, the system is learning to generalize the information in the knowledge graph by fusing it with the information present in the word embeddings. The interesting part about this is that as long as you can generate the word embedding, you will be able to generate its expanded retrofitted counterpart, and since we are using FastText(3), the generation of new out of vocabulary entries is robust thanks to the sub-word information learned in the training of FastText. Additionally, since we have learned the mapping to the retrofitted counterpart, we are no longer limited to in-knowledge entities. An example of this is if our KB did not have the entity doggo. Doggo is internet slang for dog. With RetroGAN we can generate a retrofitted embedding for doggo that should have similar information to that of the dog embedding.

After we have this retrofitting mapping through RetroGAN we run into the problem that we need to be able to extract the learned knowledge from the embeddings. This is useful, because with it we can make downstream reasoning systems and we can build knowledge bases. This is where we introduce our other contribution, a system called Deep Relationship Discovery (DRD) whose inputs are pairs of learned-retrofitted word vectors, and its outputs are the strength of commonsense assertions between the two input concepts. Intuitively, DRD learns that semantically similar entities should have similar assertions.

Putting it all together, with the combination of the RetroGAN and the Deep Relationship Discovery, if we generate a pair of word embeddings (on possibly new entities) and pass it through the RetroGAN system, we can get a an expanded commonsense-retrofitted representation of these pairs. We can then deconstruct the implicit commonsense relationship information by passing the pair through the Deep Relationship Discovery system. The result is how those two concepts relate within the context of common sense. In the case that we were exploring some new topic document, if we extract the entities in that document and we iterate over all of the pairs of entities in a new topic, then our end result is a set of assertions that show how the entities in the new topic relate from the perspective of commonsense.

In the following sections we go into details of the RetroGAN system and of the Deep Relationship Discovery system. Later on we give some background information on relevant topics and talk about

some related works. We conclude by talking about future work and directions that we intend to explore.

2 RetroGAN

2.1 System Overview

The original retrofitting operation presented previously, can only be performed on entities that exist in the knowledge graph. Intuitively this means that we can only give information about things we know about. However, we build upon the approach presented in (5) by extending it to have a CycleGAN architecture rather than a regular GAN architecture. We chose the CycleGAN architecture because it should constrain the domain of the generated embeddings more than a regular GAN because it makes sure that the generated embeddings can be transferred back to their original form. The use of this adversarial technique permits us to learn a mapping to be able to retrofit any input word embedding. Intuitively, the adversarial technique expands the KB that is used to retrofit because it tries to make similar words have similar retrofitted counterparts, even if these are not in the KB.

2.2 System Architecture

The RetroGAN system that was constructed is based on a CycleGAN architecture. On a high level, the way that CycleGANs work is that a point from a distribution X is sampled and passed to a generator network, whose output should be a point in a distribution Y. This output is checked for accuracy with a discriminator.

The cycle part comes in where the reverse process is also done: the generated sample that belongs to the distribution Y is passed through another generator network that tries to convert it to the original distribution X point. This last output is also passed to a discriminator to check that it belongs to the original distribution. As the system is trained, losses are added up as the cycle is being performed. The end result, at least in the visual domain, is a more accurate transformation than a regular GAN system as the generation process is brought full circle and constrained from both directions.

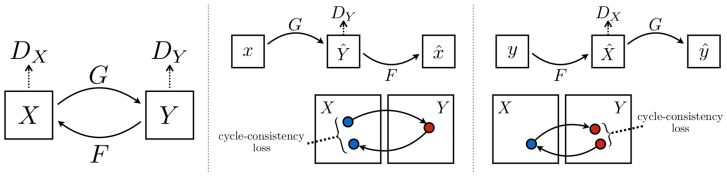

Figure 2: General CycleGAN architecture taken from (4)

Adding to this architecture, we add an attention mechanism in both of the discriminators and generators with the idea of making the system focus on key parts of the input and output vectors. This also stabilizes the training of the system(6). Additionally, in our generator we add a couple of 1D convolutional layers with the intention that these layers smoothen and filter out unimportant aspects of the input word vectors.

Our attention mechanism consists of a densely connected layer with a softmax output, that is multiplied by the input layer along with a scaling constant, that is then added to the output of the prior layer. What this does is that the important parts of the input layer are highlighted through the multiplication of the softmax layer and scaled appropriately with the constant layer which is then added to the output of the layer before the attention mechanism. The net result is that the network should give more emphasis on the relevant parts of the layer that is placed before the attention mechanism.

For each of our GAN systems the network architectures are as follows. The generators consist of a densely connected layer, followed by two 1D convolutional layers, followed by a max pooling layer, followed by a flattened layer, an attention mechanism, a densely connected layer and finally a densely connected layer that serves as the output. The discriminators are 5 densely connected layers that progressively get smaller in size with an attention mechanism in the middle. Each of our densely connected layers is followed by a batch normalization layer with the intention of stabilizing and

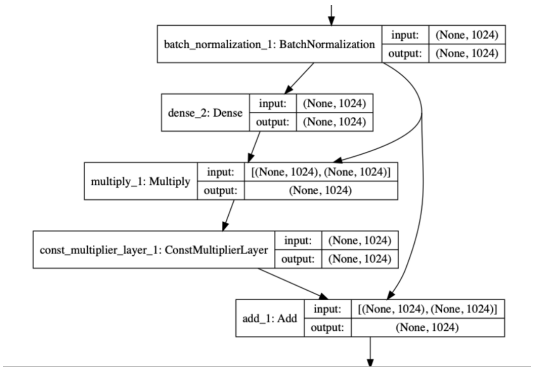

Figure 3: Our attention mechanism with a Batch normalization layer as an input

aiding in convergence of the training. Each of our layers has the ReLu(7) as an activation function except the outputs of the generator and the discriminator.

2.3 Training and Testing

To train our system, we selected the FastText word embeddings trained on the OpenSubtitles dataset(8) to serve as the input to our system and performed the retrofitting procedure utilizing ConceptNet as a KB. We selected FastText with the intention that the use of sub-word information could be used to generate out of vocabulary entities. We then proceeded to filter out by cosine distance the word embeddings that changed the most after the retrofitting procedure to use as a training set. We utilized a hardcoded threshold for this. We ran through the retrofitted procedure, and set the hardcoded threshold to the average cosine distance between the original FastText embeddings and the retrofitted embedding. The vectors that changed and had a cosine distance less than the threshold signified that they had deviated considerably from their original embeddings. We utilized these as the training set, because it meant that these embeddings were the ones that had the most connections in the knowledge graph. The ones that changed but were above the threshold meant that they had not deviated considerably and were possibly not in the knowledge graph or had very little information in the knowledge graph. We set these as the testing set. We set the number of epochs to 50, and a batch size of 32 per epoch.

Although the RetroGAN system is not the focus of this work, we tested it on the SimLex-999 (9) dataset and on the SimVerb-3500(10) dataset. Our results can be seen in table 1, along with those presented in (5).

Table 1: Spearman's correlation scores for three standard English distributional vectors spaces on English SimLex-999(SL) and SimVerb-3500 (SV). POST-DFFN uses a deep non-linear feed-forward network to learn the retrofitting mapping (11). AUXGAN is the work presented in (5), and RetroGAN is our work

Name	SL	SV
POST-DFFN	0.503	0.340
AUXGAN	**0.513**	**0.394**
RetroGAN	0.451	0.332

As we can see, our RetroGAN system is slightly inferior to the AUXGAN system presented in (5). We believe this is because of parameter tuning that is done on the AUXGAN system. It is also possible that the FastText embeddings used in the AUXGAN system are different than the ones that were utilized in our system. Another possible explanation is that the retrofitting in the AUXGAN system is the state of the art ATTRACT-REPEL(12), whereas our retrofitting is based on the original one proposed in (13).

3 Deep Relationship Discovery System

3.1 System Overview

Given that we can generate word embeddings that have imbued, generalized/expanded commonsense information in them, we now need a way to utilize that knowledge. The way we tackle this is by performing the opposite operation: we extract the assertions that are implicit in the pairs of retrofitted word vectors. Our approach to this problem is to build a Multi-Task Learning (MTL) system whose inputs are a pair of retrofitted word embeddings, and whose outputs are the strength of the relationships present in an existing knowledge base. In our specific case, it would be that the inputs to our system are going to be pairs of the vectors that are generated from the RetroGAN system, and the outputs are going to the strength of the set of relationships found in ConceptNet. In particular, the system will have a subset of the ConceptNet relationships which can be found in appendix A. We chose this subset because in training the MTL system, we do not want to bias the common body by training one task more than others and the subset we chose has a similar amount of assertions for each one of the relationships. Intuitively our system should learn to associate assertions to similar input concepts . Putting this in terms of the analogical reasoning, the system will be able to infer that if an animal desires food, then things similar to animals will desire things similar to food. It would also expand the knowledge in a KB.

3.2 System Architecture

Our Deep Relationship Discovery system can be broadly viewed in three sections: an input section, the body section, and the output section.

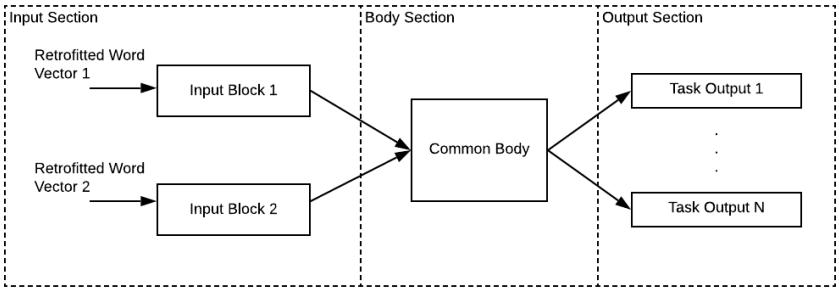

Figure 4: DRD System Architecture

The input section consists of a densely connected layer, followed by an attention mechanism, followed by another densely connected layer. The purpose of this block is to first abstract the input word vectors, then pick out the essential parts of the abstracted input, and further abstract the result of these important parts. This is done for both of the input vectors individually. The output of these blocks is then concatenated and passed to the body section.

The body section is the area that is common to all of the "tasks" that we want to predict. In our case these tasks are assertions whose weights are the strength of the relationships. The common body consists of a densely connected layer followed by an attention mechanism, followed by two more densely connected layers. The intuition behind these layers is that the pertinent information in both of the inputs is fused and then the important aspects of that fused information is highlighted. Intuitively in this area, as the system is trained, we learn the information about the knowledge base, how it is arranged and how things relate to produce certain assertions.

The output section contains a task "tail" for each of the relationships that we want to predict. We have 24 relationships that we want our system to learn. Each tail section is composed of a densely connected network followed by an attention mechanism followed by another densely connected layer, and ending with a densely connected layer for an output. The intuition for this section is that each tail takes the common body as an input and extracts the information that it needs to be able to determine the strength of the assertion that the tail represents.

3.3 Training and Preliminary Testing

The way that the Deep Relationship Discovery (DRD) system is trained is the following. We take all of the data from the assertions in ConceptNet, and proceed to filter out the data that does not contain our selected set of relationships. Then from this dataset, we proceed to sample batches of 32 assertions for each of the relationships. We then feed forward a pair of vectors from the sampled assertions, and backpropagate the difference between what our system predicts and what the actual assertion strength is. In doing this we switch between the tasks and as we train the individual tasks, we are training the common body. We run 100 epochs to train this system.

To do some preliminary evaluation of the system, we developed a graphical user interface (GUI) based on vis.js (14) with the intention of visualizing existing assertions and being able to test inference of assertions that do not exist. It is a more modern version of (15) which utilized a dimensionality reduction technique to be able to make inference on assertions.

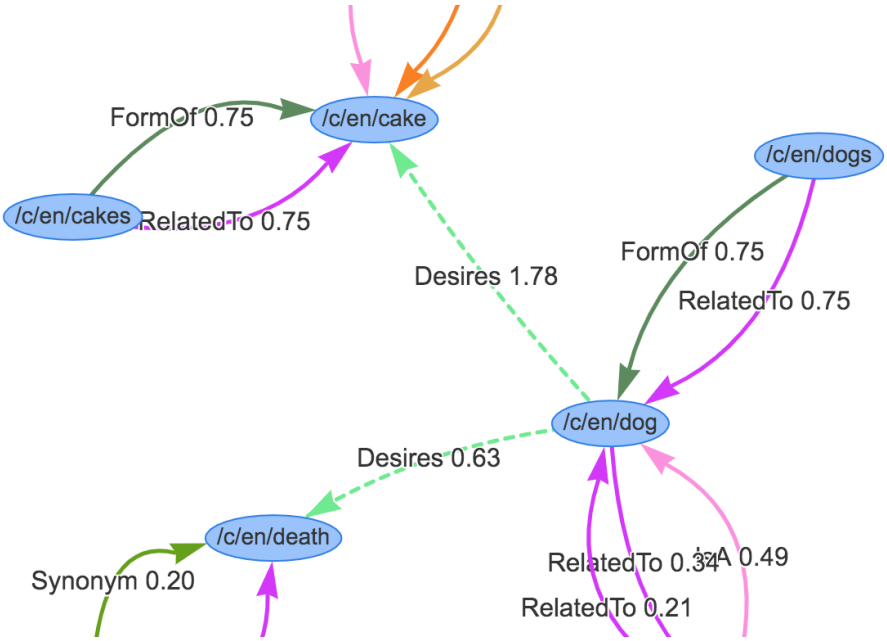

Figure 5: DRD Visualization tool. The blue ellipses are the concepts that we are visualizing, and the arrows are the the relationships between them. The solid arrows are existing relationships and the dotted arrows are inferred relationships.

The GUI lets a user supply a prompt concept, and it loads a certain amount of the nearest neighbors of that prompt concept. The GUI then proceeds to load relationships that already exist in ConceptNet and renders connections for the assertions. The GUI also lets a user input a source and destination concept and select a type of relationship to make an inference on the strength of the assertion using a backend DRD. In a simplistic example, in the GUI we can load the concept dog, the concept cake, and the concept death, and test the inference of whether a dog desires cake or if a dog desires death. This is shown in figure 5. The strength of the dog and cake assertion is shown to be 1.78 which is very different from if we tested whether a dog desires death which gives a strength of 0.63.

4 Background and Related Work

4.1 Knowledge Graphs and Knowledge Bases

A knowledge base is a set of facts, assumptions, and rules which a computer system has available to solve a problem(16). A knowledge graph is a graph structured knowledge base. In this graph, the nodes are concepts that are present in the knowledge base, the edges are the relationships between the concepts, and the weights of the edges are the strength of these relationships. An example of a small knowledge base can be seen in Figure 6.

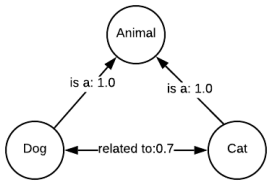

Figure 6: Simple Knowledge Graph

The way to interpret this is that a cat is an animal, a dog is an animal, and dogs and cats are related. These statements are called assertions. This graph can be interpreted as a set of assertions which take the form of tuples such as: (source concept, destination concept, relationship, strength).

4.2 A Commonsense Knowledge Base: ConceptNet

ConceptNet is a knowledge graph that connects words and phrases of natural language with labeled edges. Its knowledge is collected from many sources that include expert created resources, crowd-sourcing, and games with a purpose. (2) The idea behind ConceptNet is that systems can leverage it to understand the meaning(s) of the words that people utilize. We utilize ConceptNet in our work as the KB for concepts and assertions, because it gives a very general understanding of how the world works. Crossbridge(17) has also utilized ConceptNet to be able to make analogies with the intention of being able to perform reasoning with the analogies. However this work utilizes dimensionality reduction techniques which make it hard to evaluate out of knowledge entities.

4.3 Multi-Task Learning

Multitask learning (MTL) is the procedure of learning several tasks at the same time with the aim of mutual benefit.(18) The core idea is that the information is propagated through the common layers and then tailored to a specific output pipeline. In essence the common layers act as information that is shared throughout the tasks. MTL has been used with the purpose of constructing a KB. In the system MultiE(19), the authors propose using a MTL approach in which the inputs to the system are a set of queries (entity, relationship, ?) and the output of the system are the likelihood that an entity is the missing item in the tuple. We explore the use of MTL with a different configuration to try and leverage its common knowledge for the prediction of assertions to constructing a KB.

4.4 Word Vectors and Retrofitting

Neural word representations or more commonly known as “word embeddings” are a vectorial representation of words. This vectorial representation is extracted from layer of a neural network architecture. Word embeddings contain mostly lexical information about the corpus that they are trained on. Word embeddings also contain a small amount of semantic information. The amount and quality of semantic information in them varies greatly depending on the corpus of data used to train them, and whether there are prevalent biases in that corpus or not.

To remedy this, a procedure called retrofitting(13) was developed to help imbue word vectors with some of the semantic information of its neighbors in a knowledge graph. Retrofitting is a process in which word vectors whose entities are present in a knowledge graph, can be iterated over with the purpose of modifying them in order to make them more similar to the entities that they are connected to in a knowledge graph.

The procedure tries to strike a balance between an original word embedding and the word embeddings that are its neighbors in a knowledge graph. It is important to note that only word embeddings that are present in the knowledge graph will be modified by this equation. An improved version of this algorithm called expanded retrofitting(2), uses dimensionality reduction of multiple corpuses of retrofitted words to produce embeddings, along with the average of the embeddings around it to try and generalize and address the issue of out of vocabulary/out of knowledge graph words.

More recent work has been done in utilizing adversarial approaches to learn a mapping to be able to generalize retrofitting(5). We extend this work to use a CycleGAN(4) architecture to try and learn a more robust mapping.

5 Future Work

There are many areas that this work can be improved and continued. We intend to test our RetroGAN system by training it with Attract-Repel retrofitting strategies and evaluate it with downstream tasks such as lexical text simplification similar to what is done for AuxGAN (5) to understand better the effect of the CycleGAN architecture in learning the mapping. We intend to test our Deep Relationship Discovery system through human evaluation of previously unseen assertions. Additionally, we want to explore the optimization of the network configuration and to explore different ways to train the system by augmenting the data with some noise possibly to improve the generalization performance.

Looking at other areas, we want to leverage domain specific knowledge with general commonsense knowledge. To this end we are working on developing a transfer learning mechanism so that our system can adapt the commonsense understanding to some topic dependent knowledge. The reason for this is to leverage the connections and assertions that appear on a domain specific matter and combine it with the much broader commonsense information. If we were able to achieve this, we could build systems that can produce KBs that can be used for task-specific reasoning.

6 Conclusion

This work presents an expansion on work done to generalize retrofitting mappings though the use of a CycleGAN system called RetroGAN. Additionally, we develop a novel way to discover commonsense-based assertions between entities, by training a MTL system on a subset of the assertions present in ConceptNet. We explored the combination of the RetroGAN system with the Deep Relationship Discovery one to be able to infer assertions from concepts that may or may not be in the vocabulary, and that may or may not be in the knowledge base. We utilize this system to be able to infer that a dog does indeed desire cake!

References

[1] D. Gentner, "Structure-mapping: A theoretical framework for analogy," *Cognitive science*, vol. 7, no. 2, pp. 155–170, 1983.

[2] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[3] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[4] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[5] E. M. Ponti, I. Vulić, G. Glavaš, N. Mrkšić, and A. Korhonen, "Adversarial propagation and zero-shot cross-lingual transfer of word vector specialization," *arXiv preprint arXiv:1809.04163*, 2018.

[6] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018.

[7] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[8] J. Tiedemann, "News from opus-a collection of multilingual parallel corpora with tools and interfaces," in *Recent advances in natural language processing*, vol. 5, pp. 237–248, 2009.

[9] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics*, vol. 41, no. 4, pp. 665–695, 2015.

[10] D. Gerz, I. Vulić, F. Hill, R. Reichart, and A. Korhonen, "Simverb-3500: A large-scale evaluation set of verb similarity," *arXiv preprint arXiv:1608.00869*, 2016.

[11] G. Glavaš and I. Vulić, "Explicit retrofitting of distributional word vectors," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34–45, 2018.

[12] N. Mrkšić, I. Vulić, D. Ó Séaghdha, I. Leviant, R. Reichart, M. Gašić, A. Korhonen, and S. Young, "Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints," *Transactions of the association for Computational Linguistics*, vol. 5, pp. 309–324, 2017.

[13] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting word vectors to semantic lexicons," *arXiv preprint arXiv:1411.4166*, 2014.

[14] "Vis.js available at `https://visjs.org`," 2019.

[15] H. Lieberman and J. Henke, "Visualizing inference," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[16] "Knowledge base, definition retrieved from `https://www.lexico.com/en/definition/knowledge_base`," 2019.

[17] J. Krishnamurthy and H. Lieberman, "Crossbridge: Finding analogies using dimensionality reduction," in *2010 AAAI Fall Symposium Series*, 2010.

[18] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, ACM, 2008.

[19] Z. Zhang, F. Zhuang, Z.-Y. Niu, D. Wang, and Q. He, "Multie: Multi-task embedding for knowledge base completion," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 1715–1718, ACM, 2018.

A Subset of relationships

The subset of the relationships that we train in the Deep Relationship Discovery system are the following:

["/r/PartOf", "/r/IsA", "/r/HasA", "/r/UsedFor", "/r/CapableOf", "/r/Desires", "/r/AtLocation", "/r/HasSubevent", "/r/HasFirstSubevent", "/r/HasLastSubevent", "/r/HasPrerequisite", "/r/HasProperty", "/r/MotivatedByGoal", "/r/ObstructedBy", "/r/CreatedBy", "/r/Synonym", "/r/Causes", "/r/Antonym", "/r/DistinctFrom", "/r/DerivedFrom", "/r/SymbolOf", "/r/DefinedAs", "/r/SimilarTo", "/r/Entails"]