ACS Poster Collection (2018) 303–315 Published 12/2018; Paper 19

Query Answering by Deductive and Analogical Reasoning in a Semantic Vector Space

Douglas Summers-Stay DOUGLAS.A.SUMMERS-STAY.CIV@MAIL.MIL
Dandan Li
Peter Sutor
Adrienne Raglin
U.S. Army Research Laboratory, 2800 Powder Mill Rd. Adelphi, MD

Abstract

By embedding a knowledge graph into a semantic vector space, we are able to perform inference over a body of knowledge that can handle ambiguity, association, analogy, and abduction naturally as part of the process. We show how sparse vector decomposition can be used to find paths connecting concepts in the graph, even when such paths are incomplete. We characterize the accuracy of two main components of such a system.

1. Introduction

This paper addresses the problem of query completion with an incomplete knowledge base. A query is provided in the form $(e_1\ predicate\ X)$ or $(X\ predicate\ e_n)$ and the system attempts to find one or more X that makes the triple a true statement, along with a chain of reasoning of entities e and predicates p of the form $e_1 - p_1 - e_2 - p_2 - ...p_{n-1} - e_n$ linking e_1 to e_n that justifies the result.

When we refer to a "traditional knowledge base" in this paper, we mean a body of facts along with an inference engine capable of generating answers to queries by purely deductive reasoning. When a traditional knowledge base is missing some piece of information necessary to complete a chain of reasoning, it will return no answer at all (as discussed in Frankish & Ramsey (2014)). The facts in a knowledge base may come from different sources, which do not represent the same concept with the same terms or in quite the same way. Although each source may contain true facts, without an effort to align the representations, chains of reasoning which require facts from both sources may not go through.

In many cases it would be useful for an inference engine to come up with a best guess rather than returning no result at all. This requires a form of knowledge representation that can handle ambiguity, association, and analogy. One such representation is known as a semantic vector space, where each concept is represented by a vector, and similar concepts are represented by similar vectors. The similarity between vectors encodes subsymbolic information about their relationships. Each concept is defined by its spatial relationships with neighboring concepts.

Douglas Hofstadter described concepts in the mind in a similar fashion: "This buildup of concepts over time does not in any way establish a strict and rigid hierarchy. The dependencies are

blurry and shaded rather than precise, and there is no strict sense of higher and lower... since dependencies can be reciprocal. New concepts transform the concepts that existed prior to them, and that enabled them to come into being; in this way, newer concepts are incorporated inside their 'parents' as well as the reverse." (Hofstadter & Sander (2013))

Conceptual similarity is closely tied to the notion of analogy. When humans reason, we often use analogy to fill in the gaps. For example, precedent and analogy are used to decide cases which are not completely determined by the law (Lamond (2016)). When we encounter a new situation, we are often able to figure out how to respond by analogy with similar situations we have found ourselves in before. In automated reasoning, however, analogy is usually treated as completely separate from deduction. While analogical relationships show up in vector representations without explicitly being trained for, they are only possible to find in the purely symbolic representations used in knowledge bases when the features the analogy depends on being parallel have been explicitly assigned in the KB. (This is why, for example, it was only with the creation of vector-based analogy-finding methods that human-level performance on the SAT-analogies test/citeturney2004human set was possible.)

Section 3 lays out a form of propositional reasoning that can dependably perform inference over a set of facts, as any reasoning engine can. However, because the propositions are embedded in a vector space in an appropriate way, it can still reason in situations where it doesn't have all the facts. The answers it comes up with in these cases will be uncertain, but humans perform remarkably well in a complex environment without certainty most of the time, making it plausible, at least, that representations that allow such approximate reasoning are worth exploring.

Section 4 extends this representation beyond propositional logic to deal with predicate logic. The next few sections explain how abduction and analogy are naturally incorporated into the chains of reasoning the system generates. This is followed by a few experiments looking at how well this scales to large datasets and long chains.

2. Related Work

Representing knowledge as vectors has been explored by several relatively separate communities. The biologically-inspired cognitive architecture community has developed a method known as Vector Symbolic Architectures (VSA) (Gayler (2004)). These are capable of representing triples in a way that can be reasoned on, but to date most VSA models have been small proofs of concept. Because of the scrambling effect of the binding operators, they aren't able to do a good job of finding similarities between concepts derived from different definitions. Kanerva (1988) pioneered using sparse high-dimensional binary vectors to store knowledge in a noise-resistant way which was addressable with exemplars. Others developed these ideas to have more biologically accuracy (Widdows & Peters (2003), Knowlton et al. (2012), Levy (2009)).

The linguistics community developed Latent Semantic Analysis to create document context vectors(Dumais et al. (1988)). The ability of such vectors to solve multiple choice analogy problems at human levels was demonstrated by Turney (2005). The quality of such vectors was greatly improved by the development of the incrementally updating skip-gram method used in the well-known word vector representation word2vec (Mikolov et al. (2013)), which allowed much larger training

corpora to be used. Encoding the meaning of sentences by composing the meaning of the words in the sentence (Kiros et al. (2015), Grefenstette & Sadrzadeh (2011), Baroni & Zamparelli (2010)) is very similar to encoding triples from a knowledge base, though these authors were mainly using the vectors for finding similar meanings rather than reasoning.

TransE (Bordes et al. (2013)) and related models developed over the last few years are attempts to build a semantic vector space based on a knowledge base rather than large text corpora. TransE explictly tries to assign vectors in such a way that $source + predicate = target$ holds for all triples in the knowledge base. There have been several refinements of the TransE model to deal with the fact that representing every instance of a predicate by the same vector is really only appropriate for 1-to-1 predicates.

A few papers address multi-step deductive reasoning in semantic vector spaces, often small spaces constructed for demonstrating a capability Lee et al. (2015) Widdows & Cohen (2014) Rocktäschel & Riedel (2016) Wang et al. (2016). Neelakantan et al. (2015) for example, uses a RNN to compose steps of the path in multi-step reasoning paths. The idea of adding vectors in a chain of reasoning was used in a vector-reasoning method developed for optical devices in Caulfield et al. (2006).

Our method, developed in (Summers-Stay et al., 2016; Summers-Stay, 2017b,a), is unique in the way it incorporates both deductive and analogical reasoning, and can be used with any semantic vector space, whether derived from a text corpus (e.g. word2vec), a knowledge graph (e.g. TransE), or both.

3. Propositional Deductive Reasoning with Vectors

Before getting into the more difficult problem of predicate logic deduction with vectors, it may help to show how vectors can be used to find deductive paths in the simpler case of propositional logic.

Let each proposition literal be represented by a unique n-dimensional vector A which is not a linear combination of fewer than n of the other proposition vectors.[1] Implication $A \rightarrow B$ is represented by $-A + B$. A conjunction of literals implying a literal $A \wedge B \wedge ... \rightarrow C$ is represented by $-(A + B + ...) + C$.

We create a list of r such vectors representing all known true propositions and predicates between propositions in the model, and use these vectors to create the $r \times n$ knowledge matrix. With such a representation, it is possible to test whether any proposition P is true and find the relevant chain of reasoning by sparse decomposition of P into vectors from this matrix. Decomposition will pick out a few statements and literals to give positive weight to, and these will be the statements that participate in the proof of P. Our goal is to find a (weighted) sum of a few vectors from among a large set of possible choices. One tool to do this is known as Lasso (for Least Absolute Shrinkage and Selection Operator). Lasso was introduced in 1996 Tibshirani (1996) as an improvement on using the Moore Penrose pseudo-inverse to solve such problems by guaranteeing sparsity. Lasso is

1. These proposition vectors can be created in a variety of ways, such as skip-prop vectors (Rudinger et al. (2017)) or by combining vectors representing concepts and predicates in a Vector Symbolic Architecture (Gayler (2004)). Deductive reasoning does not put any particular constraint on the vectors used for representing propositions: a random vector would work fine for this. However, the other forms of reasoning discussed in this paper require that vectors which are nearby in vector space have similar meanings.

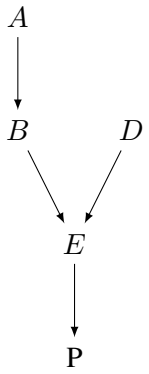

Figure 1. A tree showing implication between the propositions. The propositions that are proved true by this tree are all the nodes that participate in the tree.

a method for constraining the solution x to the problem $y = Ax + r$ where y and A are known and r is error which we would like to minimize. By making use of the L_1 norm, Lasso sets many of the elements of x to 0, which is known as a sparse solution. As a parameter in Lasso one can decide the maximum number of terms to be given non-zero weight, and the sums can be restricted to be positive.

Consider what happens when the knowledge matrix contains (among others) vectors representing the following statements and literals:

A
$A \to B$
D
$B \wedge D \to E$
$E \to P$
F
$F \to G$
...

The vector $\vec{p}$ is equal to the sum of the vectors $\vec{a}$, $\vec{d}$, $-\vec{a} + \vec{b}$, $-\vec{b} - \vec{d} + \vec{p}$, and $-\vec{e} + \vec{p}$:

$$\vec{a} + (-\vec{a} + \vec{b}) + \vec{d} + (-\vec{b} - \vec{d} + \vec{e}) + (-\vec{e} + \vec{p}) = \vec{p}$$

The vectors $\vec{f}$, $(-\vec{f} + \vec{g})$, and all other vectors in the knowledge matrix are not used in this equation and receive zero weight. This explains why the decomposition is sparse: most of the knowledge will not be used to explain the truth of a particular proposition, so most of the vectors will receive zero weight in the sum. Any vectors that receive a weight are part of the proof tree, although they are unordered, so recovering their arrangement into a tree needs to be performed by another algorithm.

The same tree can be found with less computational effort and less chance of mistakes by forward or backward chaining, as is usually done. The advantages of this kind of search are mainly in situations where some of the links in the tree are missing, as will be discussed later in the paper.

4. Predicate Deductive Reasoning with Vectors

Propositional reasoning follows chains of reasoning from proposition to proposition. Individual propositions, however, have structure as well. A simple kind of proposition links two constants, known as the source and target, by a relation known as a predicate. If a proposition is a sentence which can be either true or false, then the source, predicate, and target correspond roughly to the subject, predicate, and object of the sentence respectively. The structure of multiple constants linked by their various predicates is called a knowledge graph.

Propositional logic is unable to make the valid deduction from the two sentences "Aristotle is a man" and "men are mortal" that "Aristotle is mortal" because it treats the sentences as a whole. The deduction here requires the target of the first proposition to be recognized as the same as the source of the second proposition. In order to make the deduction, we also need the rule that "if A is a B and B is C, then A is C." This rule is known as a Horn clause, and such clauses allows us to replace a chain connecting A to C with a direct named predicate linking A to C. In general, to prove a particular predicate from A to C, only certain predicates along the chain will constitute a valid proof.[2] Knowing which predicates together constitute a valid proof of another proposition requires either creation of the Horn clauses by hand, or induction from a large knowledge base (which will have false positives when a chain holds in all known examples, but is not actually a logical necessity, a risk whenever induction is performed.)

In the method described in this paper, the Horn rules which imply a particular kind of relation limit what types of predicate links are searched. Potential chains connecting the source to the target built from these predicate links are found, and then tested to see whether they form a valid argument for a particular relation between the two according to the relevant Horn rules. For example, if we are trying to answer the query $(Claudius\ is\_the\_uncle\_of\ X)$, the system looks at the Horn rules that imply $is\_the\_uncle\_of$ and see that it needs links of the form $is\_the\_father\_of$ and $is\_the\_brother\_of$. If there were links to Claudius's father, or his father's brother, or his father's father, etc... these could also be found by the algorithm and need to be filtered out as a final step. More details about this are explained in section 8.

To represent each triple as a vector, we begin with vectors representing the constants. These vectors come from an outside source, such as distributional semantic vectors. In order to perform abductive and analogical reasoning with the vectors, the vectors must have the property that constants with similar meaning have similar vectors. To represent a particular relation between two constants, we subtract the head from the tail. All of the triples we wish to store in the vector knowledge base are turned into vectors in this way. Similarly to how we found chains of implication with propositional vectors, we will be able to find chains of predicates linking two concepts.

2. Any multi-step path through a knowledge graph is a proof of *some* proposition, though that proposition may be awkward to state. For example, if we have the propositions "birds like seeds" and "seeds grow on plants" we can represent the concepts birds, seeds, and plants as vectors, and the path birds -> like -> seeds -> grow on -> plants could be written as the proposition "seeds grow on plants and birds like them." This proposition is implied by the previous two. In some cases, the proposition proved will be expressible as a single predicate between the source and target of the chain: "Claudius is the brother of King Hamlet, who is the father of Prince Hamlet" can be expressed by "Claudius is the uncle of Prince Hamlet." These are the cases which are valid proofs of that predicate existing between the source and the target.

In a knowledge graph connecting A to C, finding such chains is relatively straightforward. There are various search algorithms (breadth-first, depth-first, single or bidirectional) to find paths in a directed graph between two vertices. However, if there is no path from A to C along the edges of the graph, such algorithms will not return any result.

Sparse vector decomposition, on the other hand, can find paths which leave the graph, if no path on the graph can be found. Normally with Lasso a single optimal path is found, but because we want to be able to select from among several possible paths, our system uses the elastic-net variant of Lasso (introduced Zou & Hastie (2005)) to find several possible paths from A to C.

Lasso will find a sum of vectors from the KB that most closely approximates the vector directly from A to C. If there is a chain of reasoning that leads from A to C by way of other entities, this sum will exactly equal the vector from A to C, and so will be the result returned. In this way, the algorithm can find a deductive path.

Finding such a path through the knowledge graph by traditional methods is not particularly difficult. The main advantage of using vectors comes from how the system behaves when there is no path from A to C at all, as discussed in the next sections.

5. Abductive Reasoning with Vectors

Suppose, however, that there is no chain of reasoning in the knowledge graph linking A to C. In a traditional knowledge base, this would be the end of the story: no results would be returned. Sparse vector decomposition, on the other hand, will return the closest possible chain of reasoning when no exact match can be found.

An incomplete chain of reasoning will have at least one gap in it, either at the beginning (proving the correct proposition, but starting from a premise which is not given) at the end (proving a true conclusion which was not the conclusion to be proved) or somewhere in the middle. In each of the three cases, the gap can be closed by hypothesizing an additional fact to fill in the gap.

It is at this point that an embedding that maps similar terms to similar vectors becomes useful. Because the gap is as minimized by the sparse vector decomposition, the terms on either end of the gap will be as closely semantically related as possible. The size of the gaps will give some indication of the likelihood that the link being abduced is a valid one.

6. Analogical Reasoning with Vectors

Any dictionary that maps similar vectors to similar concepts will be capable of finding certain kinds of simple analogies between the concepts in that dictionary. A convenient property of high-dimensional vectors is that for any pair of vectors in the dictionary, the average of those two vectors is typically closer to both of those vectors than to any of the other vectors in the dictionary. This fact allows us to look at what other concepts lie near this point, which shares semantic similarity with both of the summed vectors. Similarly, subtracting any semantic vector from another will pick out a region which is semantically similar to the positive vector while being semantically distant from the negative one. Kanerva (1988) writes,"memory items should be arranged in such a way that most items are unrelated to each other but most pairs of items can be linked by just one or two intermediate items." This property essentially allows us to find a Venn diagram for concepts

Table 1. Loosely speaking, terms near $a + b$ will come from the set of terms near a OR b, especially those terms near both a AND b, while terms near $a - b$ come from terms near a and NOT near B. Here bold terms are among the eight nearest terms to "classical" and italic terms are those near to "music".

near "classical"	classical, *classical music*, Classical, classical repertoire, Hindustani classical, contemporary, Mohiniattam, sacred choral
near "music"	music, **classical music**, jazz, Music, songs, musicians, tunes
near "music - classical"	*music*, Rhapsody subscription, ringtone, MP3s, Polow, Napster, entertainment, *Music*, *tunes*
near "music + classical"	**classical**, ***music***, ***classical music***, **jazz**, classical repertoire, **Hindustani classical**, **sacred choral**, classical guitar

semantically close to any two concepts, seeing what lies near their intersection. This is what allows simple analogies to be found.

Consider the following analogy:

$$bear : hiker :: shark : X \tag{1}$$

$Bear$ and $hiker$ both share a certain context (both are found in the forest, for example). $Bear$ and $shark$ share another context (both are predators). The vector for $bear$, then, can be decomposed into two vectors: $bear \approx forest + predator$. $Hiker$, similarly, can be decomposed into $hiker \approx forest + tourist$, and $shark$ into $shark \approx ocean + predator$. The fourth term X must share the non-forest aspect of $hiker$, and the non-predator component of $shark$, so $tourist + ocean \approx X$. A term like $snorkeler$ would share both of those contexts, so might be the nearest related term to X. This helps to make clear why the arithmetic works out, but decomposing the terms into meaningful components like $ocean$ or $predator$ isn't necessary to solve the analogy. We can simply calculate $hiker + shark - bear \approx snorkeler$, because whatever $bear$ has in common with $hiker$ and $shark$ respectively cancels out, leaving only the components of $snorkeler$:

$$(forest + tourist) + (ocean + predator) - (forest + predator) = (ocean + tourist) \tag{2}$$

Another consequence of this fact is that pairs of concepts displaced from each other by similar vectors will typically be related by the same predicate. We call these *predicate vectors*. In the case above, we might represent the predicate relating the pairs of terms by the word $threatens$: ($bear$ $threatens$ $hiker$) and ($shark$ $threatens$ $snorkeler$). Since $hiker + shark - bear \approx snorkeler$, it is also the case that $hiker - bear \approx snorkeler - shark$. In other words, the predicate vector from $hiker$ to $bear$ is approximately equal to the predicate vector from $snorkeler$ to $shark$.[3]

We should also expect that this will be more exactly true of more similar terms. Although the analogy above is valid, the predicate vectors in an analogy such as $bear : hiker :: rabid\_dog :$

3. Note that the converse of this is not, in general, true: two words can be related by the same predicate, but not have similar displacements in the vector space. The most well explored predicate for which this is the case is the hypernym-hyponym relation. Hyponyms tend to be close to, but distributed in various directions around their hypernym (Rei & Briscoe (2014)). If there were a single predicate vector representing the hypernym to hyponym relation, it would have to map all the hyponyms to the same location, which is impossible.

$jogger$ would be even more similar, since dogs are more similar to bears than sharks (claws, fur, growling, etc...) and hikers are more similar to joggers than snorkelers. At the closest, we wouldn't even call it an analogy at all, but consider both situations to be instances of the same kind, and the predicate vectors of the pairs should be nearly identical. Because analogies between vectors closer together tend to have more similar predicate vectors, we can expect the manifold formed by the predicate vectors to be locally smooth and continuous.

7. Analogical query answering with vectors

This suggests the following method for answering a query of the form (e_1 $predicate$ x) when we have many example triples that have the same predicate as our query. We can select from among these some set of n triples ($h_{1..n}$ $predicate$ $t_{1..n}$) each of whose sources h is near e_1, and do our best to approximate e_1 by a weighted sum of $h_{1..n}$: $e_1 \approx \sum_{m=1}^{n} w_m h_m$. This sum can be found by using Lasso or other sparse vector decomposition techniques. Since the space is locally smooth, the weighted average of the targets of these triples, using the same weights, should give a good estimate to the location of x:

$$x \approx \sum_{m=1}^{n} w_m t_m \tag{3}$$

We also know that x is related to e_1 in some way, so it should also be nearby e_1 in the semantic vector space. Therefore, we can give a better estimate:

$$x \approx p \sum_{m=1}^{n} w_m t_m + (1-p)e_1 \tag{4}$$

for some experimentally determined p. [4]

This approach will be particularly useful for a predicate like ($work$ $written\_by$ $person$). In general, the predicate $written\_by$ will map written works to the names of individuals. However, written works include at least two clusters: novels and movie scripts. Provided a few examples of each, our locally-linear-mapping method will find neighbors from among movies for a particular movie script and from among novels for a particular novel. In this way it is able to represent whatever internal structure a particular predicate relation may have.

8. Combined reasoning with vectors for query answering

We now have a method of locating a region in the vector space where we expect the answer to be found, and a method of finding chains connecting these potential answers back to the source entity:

1. The system is given a query of the form (e_1 r X), and its goal is to choose an X for which the query is true, and which can be proved or at least partially justified.

2. It begins with concepts already mapped into a vector space, and a knowledge matrix of predicates between those concepts.

4. The case of (x $predicate$ e_1) can be handled in the same way, only subtracting the predicate vectors instead of adding them.

3. It estimates the location of X by the method described in section 7 (forming multiple analogies with nearby examples and taking a weighted average of the analogical estimates, and restricting the answer to lie near the source vector).

4. It restricts the search to those predicates in the knowledge matrix which are known from the Horn rules to possibly participate in a proof of $(e_1\ r\ e_2)$, as discussed in section 4.

5. It finds links in the best paths from the source to the estimate of the target by performing sparse vector decomposition on the vector from the source to the target in the restricted knowledge matrix, as discussed in section 3.

6. It organizes these links into paths, and selects those that satisfy the Horn rule constraints.

9. Experiments

In order for the system to work well, both the estimate of the target location in the vector space and the path from the source to the target must be correct. We performed separate experiments on these two components to characterize their success rate.

To test the system as a whole, we will need to find a good source of Horn rules, and human judgment will be needed to decide whether each justification that doesn't constitute a full proof will be considered essentially correct. Such an experiment is still in development.

9.1 Experiment 1

We extracted 668,733 triples from ConceptNet for which both the source and target were represented by a vector in word2vec. The individual source and target concepts $h_{1..n}$, $t_{1..n}$ were thus represented by the corresponding vector in word2vec, and each predicate $(h_m\ predicate\ t_m)$ by the vector $t_m - h_m$. There were 315 many-to-many predicates in this set, with some predicates having many more examples than others.

From this set of triples we removed all 11,616 triples with a unique target and randomly selected 1000 of these unique triples to use as a test set. This ensured that the triples used for prediction would be distinct from the triples used for testing.

For each of these 1000 test triples, we used Lasso to find a weighted sum of source vectors that best approximated the source vector of the test triple. The parameter L determines the maximum number of terms with a non-zero weight in Lasso. We tested four L values: 1 ,5, 10, and 15. The other parameters for Lasso were held constant: $\lambda = 0.2$ and positive weights only.

The weights calculated for the sources were then used to create a weighted sum of target vectors from the same triples. This weighted sum of target vectors was then added to the source vector to create our estimate of the target vector of the test triple, and a weighted sum of this estimate and the target vector were used for the final estimate, with an experimentally determined weight of .67 on the source vector and a weight of .33 on the estimate. To measure accuracy, we calculated the rank of the true target vector of the test triple among the nearest neighbors of the estimated target vector. We tested with neighbors found from among two distributions: 1,000 terms randomly selected from the 1,000,000 most common word2vec terms, and 1,000 neighbors randomly selected from 30,000

Table 2. Random vectors (out of 1000)

	L=1	L=5	L=10	L=15	source	source+ mean	traditional KB
Hits@1	551	549	546	**548**	546	546	1
Hits@10	728	740	741	**743**	734	729	10

Table 3. Related vectors (out of 1000)

	L=1	L=5	L=10	L=15	source	source+ mean	traditional KB
Hits@1	260	**267**	257	266	158	4	1
Hits@10	470	478	484	**485**	298	176	10

terms near the source vector of the test triple. The second set of neighbors was chosen because these terms can be expected to be related in some way to the source term. In each case one vector was replaced with the true target vector of the test triple, so it can be considered as a multiple choice test with 999 distractors.

For comparison, we also tested results on these two test sets using the source vector from the test triple alone (source only) and on the source vector added to the mean of all examples of the predicate vector. The comparison with a traditional knowledge base is here only to emphasize that this is testing only the condition where the necessary facts do not exist in the KB, in which case the results are no better than random.

Being near the source vector is a clearly a very important factor in determining the target vector. Among vectors which are all somewhat related to the source vector, our model performs better in comparison, achieving 267 correct answers as opposed to only 158 for the nearest to the source vector. In general, the number of neighbors included in the weighted average had little effect.

Using a single mean vector to represent every instance of a predicate did very poorly by comparison, achieving only 4 direct hits and 176 times where the true answer was within the top ten results. This suggests that taking a weighted average of multiple nearby examples of a predicate might be able to improve methods such as TransE Bordes et al. (2013) which attempt to use only a single vector to represent all cases of a predicate.

Note that being the top hit, or even among the top 10 hits, is not the only criteria for whether an answer is chosen. It must also pass the test of having a path that leads back to the source without large gaps, and to count as a proof must also satisfy the Horn rules. Such restrictions allow the system to reject many false positives. This experiment simply indicates that the system is looking in the right area of the vector space.

9.2 Experiment 2

Lasso is not always able to find the path linking the source to the target, even when only one exact path exists. For this experiment we found paths of length from three to ten edges through the

Table 4. Accuracy of Lasso in finding paths of various length in the knowledge matrix

length of path	3	4	5	6	7	8	9	10
percent found	100	100	100	97	93	93	86	59

knowledge graph. In each case these were the only path connecting the source to the target. Table 4 shows, for various lengths of paths, what percent of them were found by Lasso. In the other cases, only part of the path matched the true path. Each experiment was performed a hundred times on different inputs.

This experiment was performed on a knowledge matrix formed from all the relations in Conceptnet that relate terms from among the first 50,000 in word2vec. This amounts to 43,000 relations.

A full knowledge matrix could potentially be much larger than this and achieve the same performance, since only the parts of the matrix that contain relevant predicates (as decided by the Horn rules) would be used for answering any particular query, as opposed to this test which did not remove any of the potential predicates.

10. Conclusion

Performing deductive reasoning on a knowledge graph that has been embedded in a semantic vector space seems like a promising way to include associational, analogical, and abductive reasoning into the deductive reasoning process. It allows concepts in the knowledge base to be more than just a symbolic label and include subsymbolic information relating every constant to every other constant. From the VSA point of view, using sparse vector decompostion for deductive reasoning extends the capability of VSAs in the direction of traditional knowledge bases.

We have shown that deductive reasoning can be carried out in a semantic vector space by means of sparse vector decomposition. We have characterized the ability of such a system to follow long chains of reasoning and to (sometimes) find correct answers to queries when a traditional knowledge base would not be able to answer at all. We have built a system that uses these methods to answer queries and provides approximate chains of reasoning supporting its answers.

Our next step will be to improve the validity of the proofs found by limiting the paths that can be followed to only those that would form a valid proof, and to show how the system enables two distinct knowledge bases to be used as one without finding an explicit mapping between terms.

References

Baroni, M., & Zamparelli, R. (2010). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 1183–1193). Association for Computational Linguistics.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* (pp. 2787–2795).

Caulfield, H. J., Qian, L., Vikram, C. S., Zavalin, A., Chouffani, K., Hardy, J., Mccurdy, W., & Westphal, J. (2006). Conservative optical logic devices: Cold. *Advances in Imaging and Electron Physics*, *142*, 1–52.

Dumais, S. T., Furnas, G. W., Landauer, T. K., Deerwester, S., & Harshman, R. (1988). Using latent semantic analysis to improve access to textual information. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 281–285). ACM.

Frankish, K., & Ramsey, W. M. (2014). *The Cambridge handbook of artificial intelligence*. Cambridge, MA: Cambridge University Press.

Gayler, R. W. (2004). Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. *arXiv preprint cs/0412059*.

Grefenstette, E., & Sadrzadeh, M. (2011). Experimental support for a categorical compositional distributional model of meaning. *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 1394–1404). Association for Computational Linguistics.

Hofstadter, D., & Sander, E. (2013). *Surfaces and essences: Analogy as the fuel and fire of thinking*. Basic Books.

Kanerva, P. (1988). *Sparse distributed memory*. MIT Press.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Skip-thought vectors. *Advances in neural information processing systems* (pp. 3294–3302).

Knowlton, B. J., Morrison, R. G., Hummel, J. E., & Holyoak, K. J. (2012). A neurocomputational system for relational reasoning. *Trends in cognitive sciences*, *16*, 373–381.

Lamond, G. (2016). Precedent and analogy in legal reasoning. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy*. Metaphysics Research Lab, Stanford University.

Lee, M., He, X., Yih, W.-t., Gao, J., Deng, L., & Smolensky, P. (2015). Reasoning in vector space: An exploratory study of question answering. *arXiv preprint arXiv:1511.06426*.

Levy, S. D. (2009). Distributed representation of compositional structure. In *Encyclopedia of artificial intelligence*, 514–519. IGI Global.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* (pp. 3111–3119).

Neelakantan, A., Roth, B., & Mc-Callum, A. (2015). Compositional vector space models for knowledge base inference. *2015 AAAI Spring Symposium series*.

Rei, M., & Briscoe, T. (2014). Looking for hyponyms in vector space. *Proceedings of the Eighteenth Conference on Computational Natural Language Learning* (pp. 68–77).

Rocktäschel, T., & Riedel, S. (2016). Learning knowledge base inference with neural theorem provers. *Proceedings of the Fifth Workshop on Automated Knowledge Base Construction* (pp.

45–50).

Rudinger, R., Duh, K., & Van Durme, B. (2017). Skip-Prop: Representing sentences with one vector per proposition. *Proceedings of the Twelfth International Conference on Computational Semantics (Short papers)*.

Summers-Stay, D. (2017a). Deductive and analogical reasoning on a semantically embedded knowledge graph. *Proceedings of the 2017 International Conference on Artificial General Intelligence* (pp. 112–122).

Summers-Stay, D. (2017b). Semantic vector spaces for broadening consideration of consequences. In *Autonomy and artificial intelligence: A threat or savior?*, 227–243.

Summers-Stay, D., Voss, C., & Cassidy, T. (2016). Using a distributional semantic vector space with a knowledge base for reasoning in uncertain conditions. *Biologically Inspired Cognitive Architectures*, *16*, 34–44.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, (pp. 267–288).

Turney, P. D. (2005). Measuring semantic similarity by latent relational analysis. *arXiv preprint cs/0508053*.

Wang, H., Onishi, T., Gimpel, K., & McAllester, D. (2016). Emergent predication structure in vector representations of neural readers.

Widdows, D., & Cohen, T. (2014). Reasoning with vectors: A continuous model for fast robust inference. *Logic Journal of the IGPL*, *23*, 141–173.

Widdows, D., & Peters, S. (2003). Word vectors and quantum logic: Experiments with negation and disjunction. *Mathematics of Language*, *8*.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, *67*, 301–320.