Word Embedding for Analogy Making

Mei Si

Department of Cognitive Science
Rensselaer Polytechnic Institute, Troy NY 12180, USA
sim@rpi.edu

Abstract

In recent years, natural language processing techniques have made impressive improvements in many tasks. However, their ability to make analogies is still minimal. This is partially due to the underlying representation of words and phrases, i.e., the word embedding is trained at the word sequence level and not at a concept relationship level. This work explores training a word embedding specifically for analogy making using knowledge graphs. The algorithm computes how analogous two concepts are based on the structural similarity of their adjacent concepts and relationships.

Introduction

Analogies describe comparative relationships between two sets of concepts. The Stanford Encyclopedia of Philosophy defines it as "An analogy is a comparison between two objects, or systems of objects, that highlights respects in which they are thought to be similar" (Bartha, 2019). With the recent release of large language models, such as GPT3 and BERT, natural language processing (NLP) algorithms can achieve almost human-level performance in some text generation tasks. For example, the AI Dungeon game is powered by GPT3 and can automatically generate dialogue and interactions with virtual characters as the user interact with the game. NLP algorithms have also achieved impressive performances in dialogue generation, question-answering, and even common sense reason tasks.

However, the current state-of-the-art NLP techniques still only have rudimental abilities in making analogies. A famous example of analogy-making came from Mikolov et al.'s work when the word2vec technique was invented for training word embedding (2013). Their work shows words that have similar meanings also have similar representations in the embedding space. Using vector operation, subtracting the embedding of the word Man from the embedding of King, and then adding the embedding of Women results in the embedding of Queen. I.e., the famous analogy example of:

King-Man+Women = Queen.

Another example of analogies formed based on word embedding is about locations. For example, the pair (Albuquerque, Albuquerque_Journal) is analogous to (Baltimore, Baltimore_Sun). While these analogies show that the trained word embedding is meaningful, they are not quite the same as typical analogies created by people. Further, more recent research on validating the analogies generated by word embedding found the system to be fragile and not always able to generate meaningful analogies. Even for the original example, "King-man+women" is actually closer to the embedding of King, rather than Queen (Nissim et al., 2019)!

The imperfection is not a surprise since the word2vec technique trains embedding using plain text, without exploring the relationships among concepts. In this work, we explore creating word embedding using algorithms inspired by cognitive theories of analogy, particularly the Structure-Mapping Theory (SMT) (Gentner, 1983; Gentner & Smith, 2012). SMT emphasizes the structural alignment of the relationships between two sets of concepts when forming analogies. We explore using structured content from knowledge graphs as input. The example outputs from our system show that the new embedding can create interesting and creative analogies among concepts.

Related Work

We review three types of related work: the cognitive theories about analogy-making, the knowledge graphs extracted from Wikidata, and a knowledge graph based analogy-making system.

Analogy-Making

How people form analogies has been studied extensively in cognitive science (Gentner, 1983; Kubose, Holyoak, and Hummel, 2002; Larkey and Love. 2003; Gentner and Smith, 2012). It is generally believed that analogy-making involves mapping concept groups with hierarchical structures from different domains.

The Structure-Mapping Theory (SMT) points out that analogical mapping is created by establishing a structural alignment of the relationships between two sets of concepts. The closer the structural match is, the more optimal the inferred analogy is. Surface features, i.e., properties of concepts that are not included in the hierarchical relationship structures, play little role in determining the analogy.

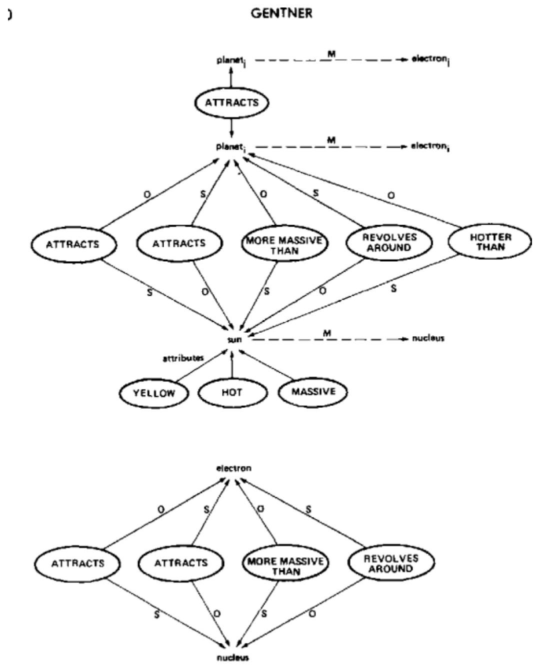

Figure 1: The analogy between the solar system and the Rutherford model (Figure taken from (Gentner, 1983).)

The Structure-Mapping Engine (SME) is a computational system that implements SMT (Falkenhainer, Forbus, and Gentner, 1989). A typical example produced by SME is the analogy between the Solar system and the Rutherford model, as shown in Figure 1. For producing this analogy, SME compares alternative ways of mapping the two groups of concepts to each other and determines that maximum structural mapping happens when the sun is mapped to the nucleus, and the planet is mapped to the electron. This mapping receives maximum support from the structural mapping of the relationships among these concepts. In the solar system, the sun and the planet have the "attracts" relationship in both ways, i.e., they both attract each other. The sun is also "more massive than" and "hotter than" the planet. The planet "revolves around" the sun. Similarly, in the Rutherford model, the electron and the nucleus have "attracts," "more massive than," and "revolves around" relationships. Furthermore, the "attracts" relationship results from both the sun and the planet having mass and gravity. The same relationship structure exists in the Rutherford model as well.

Structured Information in Knowledge Graphs

The input data -- the concepts and their relationships -- used by SME are manually designed as entities and predicates. To enable computer programs to generate analogies automatically, we also need to enable automation in generating input data. Knowledge graphs are composed of concepts connected by their relationships. They are structured data organized similarly to the manually curated data used by SME, and therefore provide a good basis for analogy generation algorithms.

Knowledge graphs cannot be directly used for computing structural mappings as in SME. Figure 2 provides an example knowledge graph crawled from Wikipedia. The main concepts from the solar system and Rutherford model analogy – sun, plant, electron, and nucleus were used as the seed nodes, and only concepts within two steps away from the seed nodes were included. The differences between Figures 1 and 2 are pretty obvious. The manually curated relationship structures only contain a limited set of entities. However, there are no natural boundaries for the groups of concepts when using knowledge graphs. This makes directly aligning two groups of concepts not feasible. Furthermore, the knowledge graph gathered from Wikipedia is less connected than the manually curated relationship structures. Typically, there is just one relationship between each pair of connected concepts. In contrast, as seen in Figure 1, there are often many relationships between a pair of concepts. In fact, these relationships are important supporting evidence when aligning the solar system and the Rutherford model.

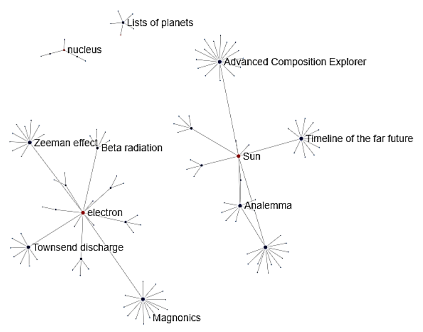

Figure 2: Sun and related concepts in Wikipedia.

Make Analogies using Knowledge Graph

This work is inspired by and based on (Si & Carlson, 2017), which uses information from DBpedia as the base for generating analogies. Si and Carlson's approach was inspired by the Structural Mapping Theory (SMT). The algorithm finds analogous relationship pairs, and the analogies are composed of a pair of mapping concepts and a set of supporting evidence, i.e., analogous relationship pairs.

An essential step in the algorithm is inferring pairs of analogous relationships. The algorithm computes how analogous two relationships are based on the topological similarity of their adjacent concepts and relationships. Si and Carlson compute four sets of relationship differences between the linked-from concept and the targeting concept:

1. Gain – what relationships are associated with the targeting concept but not the linked-from concept;
2. Loss – what relationships are associated with the linked-from concept but not the targeting concept;
3. Same – what relationships are associated with both the targeting concept and the linked-from concept;
4. Diff – the combination of the gain and the loss sets.

The differences among these sets are used to generate a unique index (embedding) for each relationship (Si & Carlson, 2017).

This relationship embedding serves as the basis for constructing analogies. If two concepts have many relationships that are analogous/similar to each other, the two concepts are regarded as being analogous. For example, Punk Rock is analogous to LPC (a programming language) because "the stylistic origin of Punk Rock is Garage Rock, Glam Rock, and Surf Music, just like LPC is influenced by Lisp, Perl, and C," and "Punk Rock is a music fusion genre of Celtic Punk, just like LPC influences Pike." Here, the analogy between Punk Rock and LPC is supported by mapping the "stylistic origin" of a music genre to the "influenced by" relationship among programming languages, and the "fusion genre" relationship among music genres to the "influence" relationship among programming languages. This approach mimics how structural mapping works in a weaker form.

Approach

This work explores an alternative approach for computing the embedding of relationships. Because the word2vec algorithm has been widely used for creating word embedding (Mikolov et al., 2013), we propose an algorithm that uses word2vec to compute the relationships embedding.

Our proposed approach contains three main steps, as illustrated in Figure 3. It first constructs a knowledge graph by crawling information from Wikidata. We use Wikidata instead of DBpedia to construct the knowledge graph.

For computing word embedding using word2vec, the words must appear in the input data many times. Only then the word2vec algorithm can learn their relationships with nearby words. Unfortunately, most concepts in Wikidata are unique, i.e., there is only one entry for each concept. Therefore, the word2vec algorithm cannot be directly applied. On the other hand, the relationships in Wikidata are rarely unique. E.g., "Give Name" is a popular relationship that connects many pairs of concepts. Therefore, in the second step, we construct a reversed knowledge graph where the relationships are nodes and the concepts are edges, as shown in Figure 4. And finally, we compute the embedding for the relationships using this reversed knowledge graph.

Construct Knowledge Graph

For getting information from Wikidata, we used a web crawler, which stores concepts and their relationships in a network structure. For creating the knowledge graph we used in this work, we used 18 seed words, and did a breadth-first search around each of them until at least 1000 nodes had been reached. Then we merged all the data collected. The resulting knowledge graph contains 219691 entities and 1540 unique relationships.

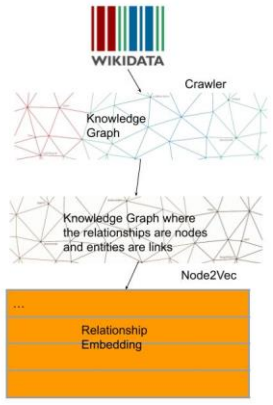
WIKIDATA
Crawler
Knowledge Graph
Knowledge Graph where the relationships are nodes and entities are links
Node2Vec
...
Relationship Embedding

Figure 3: Workflow.

For computing the embedding for the relationship, we built a reversed graph where the relationships are nodes, and the entities are links. For example, in Wikidata, "member of political party" is the relationship between "Armen Sarkissian" and "independent politician." In this reversed graph, relationships such as "member of political party" and "given name" become nodes, and the entities become edges. We then apply the node2vec algorithm on this graph to obtain the embedding for the relationships (Grover & Leskovec, 2016).

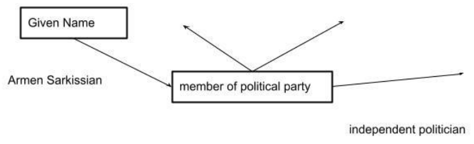

Figure 4: Reversed Knowledge Graph.

Node2vec

Node2vec is an embedding algorithm developed by Grover & Leskovec (2016). This algorithm can convert nodes in a graph into numerical representations, i.e., embedding. Node2vec works in two steps. The first step uses a second-order random walk on the graph to generate transaction samples. These samples are equivalent to the text input to word2vec, and the second step uses word2vec to compute the embedding. Take Figure 4, for example; the random walk algorithm would visit each note multiple times, and randomly follow a link to move to the next node each time. After sampling, the graph is essentially converted to a list of linear transactions, each of them contains a list of nodes, e.g. [given name, member of political party …]. These linear transactions become the corpus for word2vec.

Example Output

Like regular word embedding, the relationship embedding computed in this work allows us to calculate the distance between two relationships and find the most similar relationships. We also implemented the algorithm from (Si & Carlson, 2017) and compared these two embeddings.

Both embeddings are not perfect but can provide some insightful results. Moreover, their results read more like figurative language than a simple word association. For example, Tables 1 and 2 list the top 10 closest relationships to two relationships we used for testing. The closest ones are on the top.

Table 1: Results for "member of political party".

(Si & Carlson, 2017)	Node2Vec
Work location	Family name
Place of birth	Military rank
Place of death	Position held
Language used	Military branch
Official Language	Sibling
Residence	Spouse
Place of burial	Moth
Educated at	Native language
Parent astronomical body	Educated at
Country of citizenship	Sex or gender

Table 2: Results for "architectural style".

(Si & Carlson, 2017)	Node2Vec
Origin of the watercourse	Architect
Director/Manager	Heritage designation located on street
Material used	Drainage basin
Occupant	Material used
Lyrics by	Legal form
Anthem	Located on terrain feature
Legislative body	Located in time zone mouth of the watercourse
Legal form	Contain settlement
Currency	
Industry	

In Table 1, both embeddings suggest "Educated at" could be an analogy to "member of political party." And in Table 2, both suggest "Legal form" could be an analogy to "architectural style." We think these suggestions are pretty creative.

Note that compared to WikiData itself, our crawled dataset is tiny and sparse. Therefore, these suggested relationships are not necessarily the best analogies from people's points of view. Nevertheless, most proposed relationships convey meaning more or less similar to the source concept.

Discussion and Future Work

We aim to create analogies where the relationship mapping itself is analogous. Though the process of computing how analogous two relationships are to each other leverages the idea of computing structural similarity, we suspect the results presented here are different from results produced by SME or other systems that infer analogies solely based on structural similarities. Using SME, the symbolic meanings of the relationships are discarded, and only the structural alignment between the two groups of concepts is considered. Two relationships both named involving do not make them more analogous to each other than two relationships with different names. In our results, the meanings of the relationships are undoubtedly important. We plan to explore this phenomenon and exam further to what degree the embedding we computed is independent of the relationships' symbolic meanings in the future.

The current work finds analogous relationships, but does not use them to find analogous concepts yet. We will explore this direction in future work. We are also interested in computing the relationship embedding using a larger knowledge graph and seeing whether that improves the results.

Author Contributions

M Si ideated and wrote the paper alone.

Acknowledgements

The author wants to thank the anonymous reviewers for their thoughtful comments and suggestions.

References

Bartha, Paul: Analogy and analogical reasoning. In: The Stanford Encyclopedia of Philosophy. Spring 2019 ed. Edward N. Zalta (ed.), forthcoming URL = https://plato.stanford.edu/archives/spr2019/entries/reasoning-analogy/

Falkenhainer, B., Forbus, K., Gentner, D.: The structure-mapping engine: Algorithm and examples. Artificial Intelligence, 41, 1–63. (1989).

Forbus, K., Oblinger, D.: Making SME greedy and pragmatic. In: Proceedings of the 12th Annual Conference of the Cognitive Science Society, 61–68. (1990)

Gentner, D., & Markman, A. B.: Structure mapping in analogy and similarity. American Psychologist, 52, 45-56. (1997).

Gentner, D., Smith, L.: Analogical reasoning. In V. Ramachandran (Ed.), Encyclopedia of human behavior. 2nd ed. pp. 130–136. Elsevier; Oxford, UK. (2012).

Gentner, D.: Structure-mapping: A theoretical framework for analogy. Cognitive Science, 7 (2), 155–170. (1983).

Grover, A., & Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 855-864).

Kubose, T. T, Holyoak, K. J, Hummel, J. E.: The role of textual coherence in incremental analogical mapping. Journal of memory and language, 47(3), 407-435. (2002).

Larkey, L. B., Love, B. C. CAB: Connectionist analogy builder. Cognitive Science, 27(5), 781-794. (2003).

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

Nissim, M., van Noord, R., & van der Goot, R. (2020). Fair is better than sensational: Man is to doctor as woman is to doctor. Computational Linguistics, 46(2), 487-497.

Si, M., Carlson, C.: A Data-Driven Approach for Making Analogies. In: Proc. Cognitive Science Society Conference, pp. 3155-3160. (2017).