A Uniform Model for Generative and Discriminative Commonsense Knowledge Tuples

Harm Hoogeveen
Supervisor(s): Gaole He, Ujwal Gadiraju
EEMCS, Delft University of Technology, The Netherlands

June 19, 2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

Commonsense knowledge plays a key role in human intelligence. It is knowledge possessed by most humans that helps them in everyday situations. One possible way is to store the knowledge in four types. Each piece is either positive or negative, and generative or discriminative. For efficient retrieval and storage, a uniform model is needed. Existing models for commonsense knowledge are not fit for negative and discriminative knowledge. The aim of this paper is to create a uniform model to store both positive and negative generative and discriminative knowledge tuples. Models are evaluated on a set of generalized queries as well as on the storage they require. Four possible models were evaluated of which two were the most promising: the generative model and the combined model. The generative model is efficient in storage and retrieving generative knowledge for concepts, but relatively slow in distinguishing concepts. Combining the generative model with discriminative tuples gives the combined model, a model that is the most efficient for all queries but expensive in storage. Which of the two models is most suitable depends on the application and the available resources.

1 Introduction

Commonsense knowledge is information possessed by most humans that helps them make sense of everyday situations [1], [2] The grass is green, an apple can be eaten, and a ball is circular in shape. Commonsense knowledge comes in many types. There is generative knowledge which gives commonsense knowledge on a concept and a relation [3]. For example *an apple is a fruit* is generative knowledge. Apart from generative knowledge there is also discriminative knowledge which tells whether two concepts can be differentiated on a relation. An example of discriminative knowledge is *A dog is an animal and a car is not*, we can differentiate a dog and a car here. Apart from generative and discriminative knowledge a distinction can also be made between positive and negative knowledge. The examples above are both of positive knowledge. However, the discriminative example already shows that just having positive knowledge isn't enough [4]. The part *A car is not an animal* is an example of negative (generative) knowledge. It states that a relation does not hold for a concept. Without negative knowledge it is not known whether a relation does not hold for a concept or whether it is undefined (missing) [5]. Negative discriminative knowledge is defined as the case were two concepts can't be differentiated for a certain relation. An example of that: *A pear is a fruit and so is an apple*, the two concepts can't be differentiated on the relation that something is a fruit since both are. Each piece of generative or discriminative knowledge is either positive or negative.

An important task of AI systems is to quickly query commonsense knowledge. To facilitate this, commonsense knowledge needs to be stored efficiently so that minimal storage space is used whilst queries can be executed at maximum speed. Existing data structures for commonsense knowledge are not fit for

negative knowledge and discriminative knowledge. Therefore there is need for a model to support these types of knowledge. This paper will develop a uniform model to represent both positive and negative generative and discriminative knowledge.

In the next section, some preliminary knowledge will be given about commonsense knowledge as well as an existing tuple structure that will be used throughout the paper. After that in section 3 it will be explained how a model will be developed and how one is evaluated. Before the models are given a set of queries will be defined in section 4 where the models will later be tested on. In section 5 possible models will be created. Using the queries of section 4 the models will be evaluated in section 6. After the evaluation of each model, the models will be compared in section 7. Finally in section 8 conclusions will be made and possibilities for future work are given.

2 Preliminary Knowledge

Commonsense knowledge is used for commonsense reasoning, which is the process of interpreting and making assumptions about everyday situations [6], [7]. For example when a person grabs some food one could reasonably argue the food is going to be eaten. Because of the implicit characteristic of commonsense knowledge, it is difficult to acquire for machines. This task together with representing the knowledge and reasoning with it is a major continuous challenge in AI [1], [8]. In encyclopedic knowledge machines do not have problems, ask them about a term and they can easily retrieve much information about that term [9]. However, asking a machine "the simple task" to distinguish between a truck and an overpass it may lag; leading to fatal incidents and thus illustrating the need for commonsense knowledge so that machines are able to better distinguish different things [9]. More specifically there is a need for more positive discriminative (and thus negative generative) so that AI machines perform better at discriminating concepts [10]. Commonsense knowledge can be classified into different types and taxonomies. In-depth characterizations of knowledge have not yet been given [11].

Knowledge engineering is the research area of developing methods to gather commonsense knowledge [12]. For example, this can be done by interrogating humans through games with a purpose (GWAP) [13]. One such game is the game FindItOut! [11]. In the game, which is based on "Guess Who!" each player is presented with a set of cards containing random concepts and a single card containing the concept their opponent has to guess. Alternating, they have to ask the other player a question so they can remove the concepts that do not match the question's answer. The player who first guesses the other's card wins.

The game generates commonsense knowledge data in the form of tuples which are classified into two dimensions. Tuples are either generative or discriminative and either positive or negative. Positive-generative and negative-generative tuples take the form of +/-<concept, relation, input> where the relation and input apply to the concept and the sign says whether it is correct

or not. For example, to say an apple is a fruit the tuple would become +<apple, isA, fruit>. Discriminative knowledge tuples consist of two concepts as well as a relation and an input, and have a positive sign if the two concepts can be distinguished using the relation and input, and a negative sign otherwise. For example, an apple and a banana can't be distinguished on the relation and input isA fruit, and will therefore be in a negative discriminative tuple: -<apple, banana, isA, fruit>. If a discriminative tuple is positive the relation and input hold for the first concept [14]. This tuple structure defined by Balayn, He, Hu, *et al.* [11] will be used as a basis for this paper.

3 Methodology

In this section the process of developing possible models will be explained as well as the method that will be used to evaluate the models.

3.1 Model creation

The target of creating a model is to define a tuple structure which can be trivially used in database systems. Not all tuples have to be of the same type. A possible structure could for example consists of both triples and quadruples. The data within a single type should however be consistent. The possible models will be created using the tuple structure of Balayn, He, Hu, *et al.* [11] combined with intuitive ideas.

3.2 Evaluating the models

To create a possible model one needs to know how it can be evaluated. In this paper, the focus for evaluation will be on both time complexity and storage. For the time complexity, a set of generalized queries will be created. Every query will be evaluated for every model. The time complexity will be given in the Big-Oh notation. Because the worst-case time complexity is likely to not be relevant (this would be given for example if all relations are related to a single concept), the average time complexity will be used. For the storage usage of a model, the number of tuples needed to store a model will be used.

4 Queries

To later evaluate the models that will be given in section 5, a set of queries is necessary to find the efficiency of the models. For the definition of queries a set of queries for both generative knowledge and discriminative knowledge will be defined. The queries are chosen intuitively based on the generative and discriminative knowledge tuples. It is assumed that for every query the concept(s) on which the query applies is known.

4.1 Generative queries

For generative knowledge, the most important queries consist of getting information about a certain concept. The first two queries query the characteristics of a concept: what relations and inputs are associated with a concept. The third query is used to check if a specified sign, relation and input apply to a concept.

1. Given a concept and a sign, what are the relations and inputs?
2. Given a concept, relation and a sign, what are the inputs?
3. Given a concept, relation, sign and input does it exist?

As can be seen, some of the queries can be expressed in some other of these queries. For example query 3 can easily be obtained from query 1 with an extra check whether the inputs of query 3 are in the result set of 1. However, since some models might be able to outperform this trivial way of substituting queries they are included in the evaluation set.

4.2 Discriminative queries

For the discriminative knowledge, the first two queries request the differences between two concepts. For the sixth query, the relation and input are known and the question is whether two concepts can be differentiated by this.

4. On what relation-input combinations do two concepts differ?
5. On what relation-input combinations don't two concepts differ?
6. Can we differ two concepts for a specific relation and input?

Also for the discriminative queries it is the case that they could be written in terms of other queries with an extra check. But again some models might be more efficient than that.

5 Models

After having defined the queries, in this section, some possible models will be given where the queries are later evaluated on. For every model, a tuple structure will be given as well as a visualization in the form of a graph. For the visualizations, a small set of sample data is used which can be seen in Table 1. The source for the graph generation can be found on GitHub[1].

[1] https://github.com/HarmHoog/CS-Graph-Generator

apple	isA	fruit
pear	isA	fruit
fruit	isA	food
apple	isA	fruit
dog	isA	animal
fruit	contains	vitamins
dog	isA	pet
dog	eats	food
pet	atLocation	home
car	has	wheels
car	has	horn
human	eats	food
human	drives	car

pet	isA	food
dog	isA	food
dog	drives	car
car	atLocation	home
dog	has	wheels
human	has	wheels
fruit	has	wheels

Table 1: Sample data for the model visualizations, positive tuples are in the left table and negative tuples in the right

5.1 Generative model

The first model that will be discussed is the generative model. This model is based on the tuples discussed by Balayn, He, Hu, *et al.* [11]. In this case, only the generative tuples are used. Note that the discriminative tuples can be retrieved from combinations of generative tuples. For example if two concepts have the same sign, relation and input a negative discriminative tuple can be formed. The same goes for positive discriminative tuples except for that the sign has to be different. The tuples for this model thus look like $< sign, concept, relation, input >$. In figure 1 the Generative model is shown in a graph.

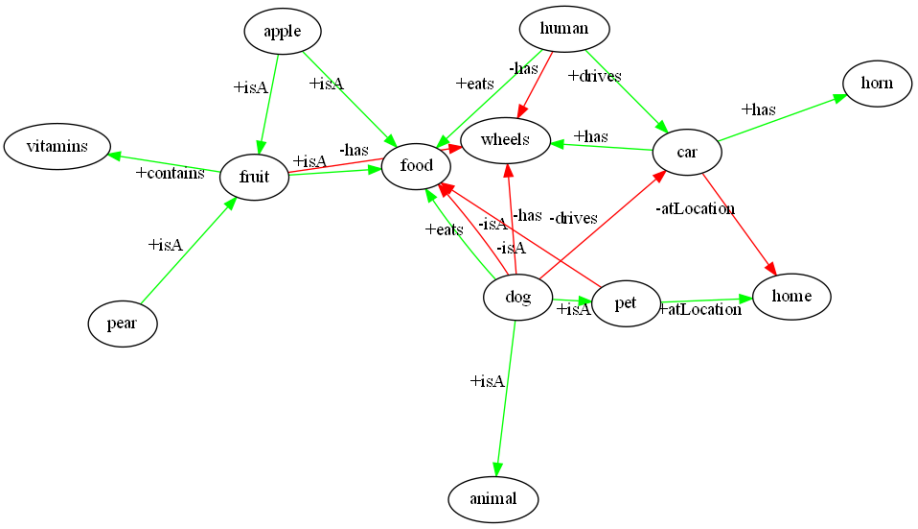

Figure 1: The Generative model visualized. Green arrows represent positive signs and red arrows negative. The direction of the arrow is given by the relation.

5.2 Discriminative model

Just like generative tuples, discriminative tuples as described by Balayn, He, Hu, *et al.* [11] can also be used to store the knowledge. Following the example of the previous section the tuples for this model look like:
$< sign, concept\#1, concept\#2, relation, input >$. If the sign is positive (e.g. we can discriminate) then the first concept is the one corresponding to the positive of the relation and input. In figure 2 a visualization of this model is given. Note that the amount of nodes in this visualization is clearly less than the amount seen in figure 1 of the Generative model. This is because discriminative knowledge is dependent on combinations of generative knowledge, and if such combinations aren't available no discriminative knowledge is available.

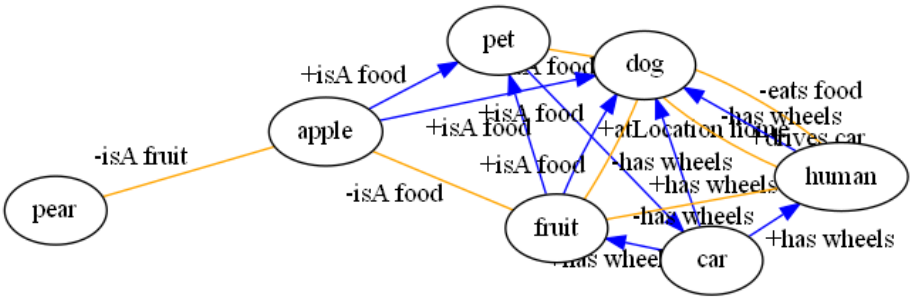

Figure 2: The Discriminative model visualized. The blue edges represent positive discriminative knowledge and the orange edges the negative. Only the positive edges are directional since for the negative edges both concepts have the same sign for the corresponding relation and input.

5.3 Combined model

Another possibility is to combine the two previous models. Instead of using one type of tuple two types are then used. Both the generative:
$< sign, concept, relation, input >$ tuples as well as the discriminative:
$< sign, concept\#1, concept\#2, relation, input >$ tuples are then used. In figure 3 a visualization can be seen.

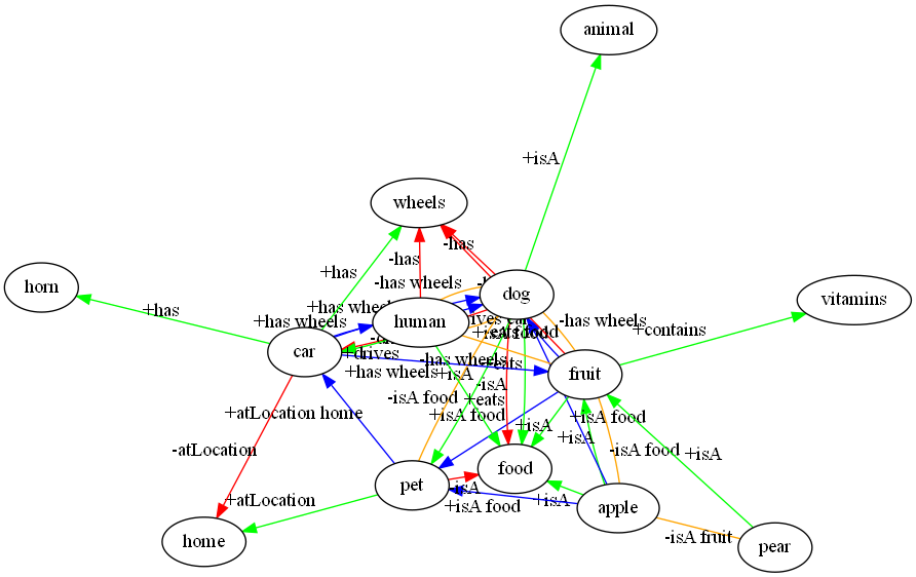

Figure 3: The Combined model visualized. The edges follow the same structure as in figure 1 and 2

5.4 Hypergraph model

Another possible optimization can be given by combining the edges of the generative model. This could be done by using so-called hypergraphs which consist of hyperedges. Each hyperedge in a hypergraph can contain multiple vertices as start and end point [15]. If in a normal directed graph multiple vertices point to the same vertex, in a hypergraph this could be combined into one hyperedge.

Using a hypergraph for the generative knowledge, tuples can be combined by using the tuple form: $< sign, relation, input, Set(Concept) >$. Every concept that has a generative tuple with the same sign, relation and input as another is then combined into a single edge. In figure 4 the hyperedge form can be seen.

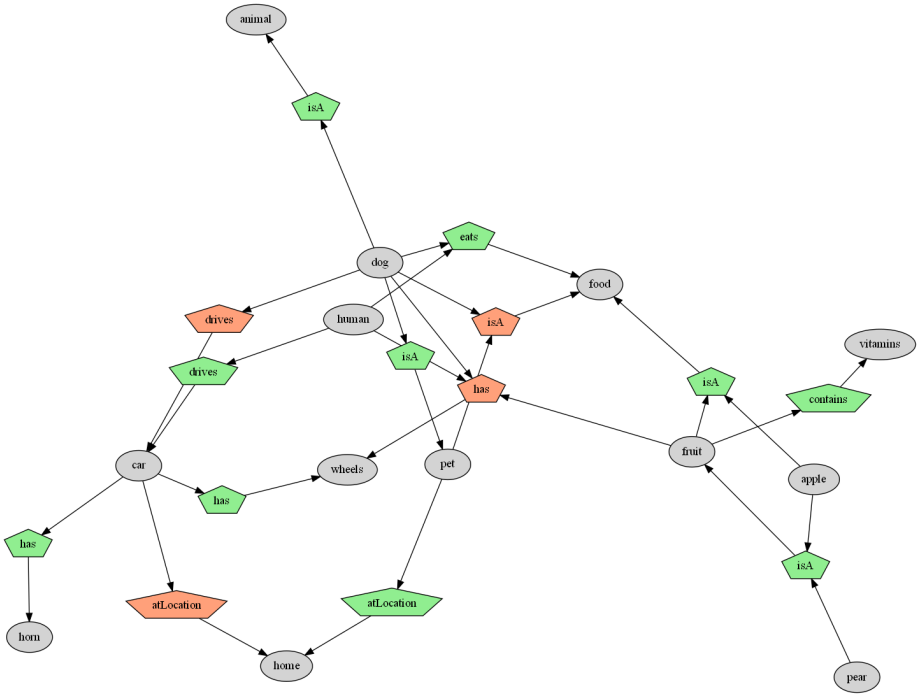

Figure 4: The Hypergraph model visualized. Every pentagon represents a hyperedge where all the incoming edges are in the $Set(Concept)$ and the one outgoing edge points to the *input*. The color of pentagon corresponds to its sign.

6 Analyzing the models

Since for all the models the tuples are based on the existing structure given by Balayn, He, Hu, *et al.* [11] the following sets will be used:
G : the set of generative tuples

D : the set of discriminative tuples
H : the set of hyperedge tuples

Because the interest is on the average time complexity the following constants are also defined:
E : The average number of generative edges a node has
L : The average number of discriminative edges a node has
A : The average number of concepts in the set of a hyperedge.
N : The number of concepts

For every model, the time complexity will be given for all queries defined in section 4. One important factor that needs to be dealt with is the time it takes to retrieve a tuple. This time is dependent on the database system that is used and the way a tuple is retrieved. For example for a set of n tuples, this could take $O(N)$ time if it just goes through all the tuples, $O(logn)$ time if a system uses for example a binary search system or even $O(1)$ if some sort of hashing is used. In this paper, it is assumed that retrieving one or more tuples from a database using its key, which will be defined for every model, takes $O(1)$ time. The choice of using the $O(1)$ time was made because then the query time in Big-Oh does not depend on the tuple retrieval and therefore the query execution time can be more explicitly stated. In the generative model, the key is the concept and in the discriminative model, it is either one of the concepts. More explicitly: getting all generative tuples of a single concept takes $O(1)$ time. The same goes for the discriminative model but then for both concepts. For the combined model a combination of the two applies. The only model that is slightly different is the hypergraph model. The hypergraph model uses the sign, relation and input combined as a key to retrieve tuples.

6.1 Generative model

Given that retrieving a set of tuples related to a concept is done in $O(1)$ time the time complexities for the generative model are mostly trivial. For the first three queries, the tuples are retrieved for a concept and filtered in $O(E)$ time. For the next two queries, it is necessary to retrieve the tuples of both concepts and find combinations of the two which can be done in $O(E^2)$ time. For the last query for both concepts, it needs to be checked whether the relation and input hold which can be done in $O(E + E)$ time. Since the number of tuples is the same as the number of generative tuples the storage is $|G|$.

QUERY	TIME
Given a concept and a sign, what are the relations and inputs?	$O(E)$
Given a concept, relation and a sign, what are the inputs?	$O(E)$
Given a concept, relation, sign and input does it exist?	$O(E)$
On what relation-input's do two concepts differ?	$O(E^2)$
On what relation-input's don't two concepts differ?	$O(E^2)$
Can we differ two concepts for a specific relation and input?	$O(E+E)$

Table 2: time complexity results for the generative model

6.2 Discriminative model

Finding the time complexities for the discriminative model is more complex. For the first three queries, the problem arises that not all knowledge can be retrieved using only discriminative tuples. This can easily be seen from the negative discriminative tuples. For two concepts it is known that in terms of generative knowledge the combination of the relation and input is either positive or negative for both concepts but it is unknown which it is. Therefore the first three queries can't be executed on the discriminative model. For the last three queries, the time complexity is more trivial in terms of the number of edges, it consists of looping through the discriminative tuples, that a concept has, and filtering out the right ones. Therefore the time complexity for these queries is $O(L)$. In an ideal situation, the constant L could be compared to the number of generative edges E. This is the case when for all tuples of generative knowledge every concept has either a positive or negative equivalent of that tuple. Then there exists a tuple of discriminative knowledge for every relation and input between all concepts. Therefore L can be at a maximum $n * E$ (where n is the number of concepts).

Just like the generative model the storage for the discriminative model is $|D|$ since only discriminative tuples are used. The amount of discriminative tuples is usually very high compared to the number of generative tuples. This is because one generative tuple can be used to create $N-1$ discriminative tuples if there exists a generative tuple with the same relation and input for every other concept.

QUERY	TIME
Given a concept and a sign, what are the relations and inputs?	-
Given a concept, relation and a sign, what are the inputs?	-
Given a concept, relation, sign and input does it exist?	-
On what relation-input's do two concepts differ?	$O(L)$
On what relation-input's don't two concepts differ?	$O(L)$
Can we differ two concepts for a specific relation and input?	$O(L)$

Table 3: time complexity results for the discriminative model

6.3 Combined model

Since the combined model uses tuples from both the generative and discriminative models, the best of both the models can be selected. The discriminative model is not conclusive for the first three queries so the results of the generative model will be selected. For the last three queries, it has to be known whether $L < E^2$. Because not all combinations of generative knowledge between concepts are defined on average it can be assumed this is indeed the case. Therefore for the last three queries, the discriminative model will be selected. For the storage, it uses both the generative and discriminative tuples so this will be equal to $|G| + |D|$.

QUERY	TIME
Given a concept and a sign, what are the relations and inputs?	$O(E)$
Given a concept, relation and a sign, what are the inputs?	$O(E)$
Given a concept, relation, sign and input does it exist?	$O(E)$
On what relation-input's do two concepts differ?	$O(L)$
On what relation-input's don't two concepts differ?	$O(L)$
Can we differ two concepts for a specific relation and input?	$O(L)$

Table 4: time complexity results for the combined model

6.4 Hypergraph model

For the hypergraph model, the tuples use a different structure and because of that, the queries are processed much differently. For the first two queries, there has to be looped over all the hyperedges and included concepts which will take $O(HA)$ (where H is the number of hyperedges and A the average number of concepts per hyperedge) time. For the third query the hyperedge is known and only the set of concepts needs to be checked in $O(A)$ time. For the fourth and fifth query for every hyperedge, it needs to be checked whether the second concept exists also in either the same hyperedge or in the equivalent hyperedge with only a different sign. This can be done in $O(HA)$. For the last query the specific hyperedge is known and therefore only the set of concepts needs to be checked.

The storage in terms of the number of tuples is logically much smaller than the previous models since all relations and inputs are combined. The tuples on themselves are larger now since they store a set of concepts. It is comparable to the generative knowledge in storage however every combination of sign, relation and input is only stored once. Therefore the storage of the hypergraph model is less than $|G|$.

QUERY	TIME
Given a concept and a sign, what are the relations and inputs?	$O(HA)$
Given a concept, relation and a sign, what are the inputs?	$O(HA)$
Given a concept, relation, sign and input does it exist?	$O(A)$
On what relation-input's do two concepts differ?	$O(HA)$
On what relation-input's don't two concepts differ?	$O(HA)$
Can we differ two concepts for a specific relation and input?	$O(A)$

Table 5: Time-complexity results for the hypergraph model

7 Comparison and Discussion

Considering that the discriminative model cannot execute the first three queries it will not be further considered. It might only be suitable for applications where the sole purpose is to distinguish concepts, but even then the combined model will give the same results for slightly more storage (the set of discriminative tuples is usually much larger than that of the generative).

For the other three models, there seems to be a trade-off between storage and query time. The generative and hypergraph models are using less storage than the combined model but are less efficient in query execution. Because the checking of every hyperedge and associated concepts is an expensive operation the generative model outperforms the hyperedge model in query execution whilst the hyperedge model only uses slightly less storage. This makes the generative model preferable.

Between the generative model and the combined model, the choice needs to be made whether one cares more about query execution speed or storage. The generative model uses little storage compared to the combined model. However, the combined model is better in distinguishing concepts. Which one is more suitable depends on the application and the available resources.

8 Conclusion and Future Work

The goal of the paper was to create a uniform model for generative and discriminative commonsense knowledge tuples. The research has resulted in two possible models: the generative model and the combined model. The generative model is efficient in storage but relatively slow in executing discriminative queries whilst the combined model is the fastest in query execution but relatively more expensive in storage. Which one is more suitable depends on the application as well as on the available resources.

Another solution that might be worth looking into in future research is a complex model. Using graphs with generative knowledge on the axis, discriminative knowledge could be represented as combinations of generative knowledge. Such a solution requires a more mathematical basis to analyze. For the hyperedge model, one could also consider using different types of edges where for

example the relations and inputs are accumulated in one edge for a concept instead of the other way around.

In conclusion, it can be said that the best model depends mostly on the application as on the available resources. Both the generative and combined model are good candidates for a uniform model and which has to be used can be best decided on which suits best for a specific application.

9 Responsible Research

The research is done in a way such that it is reproducible. Before the creation of the models a background on the tuples and knowledge base has been given on which the models were going to be based. The possible models were thoroughly explained as well as the manner in which the models were evaluated. The data and code for the graphs have been made available for reproducibility as well. The models were then analyzed on the evaluation methods and compared to each other. Finally, conclusions have been made and explained.

References

[1] F. Ilievski, A. Oltramari, K. X. Ma, B. Zhang, D. L. McGuinness, and P. Szekely, "Dimensions of commonsense knowledge," *Knowledge-based Systems*, vol. 229, p. 107 347, 2021, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2021.107347.

[2] H. Lieberman, H. Liu, P. Singh, and B. Barry, "Beating common sense into interactive applications," *AI Magazine*, vol. 25, no. 4, p. 63, Dec. 2004. DOI: 10.1609/aimag.v25i4.1785. [Online]. Available: https://ojs.aaai.org/index.php/aimagazine/article/view/1785.

[3] D. Ghosal, N. Majumder, A. Gelbukh, R. Mihalcea, and S. Poria, *COSMIC: COmmonSense knowledge for eMotion Identification in Conversations*, 2020. DOI: 10.18653/v1/2020.findings-emnlp.224.

[4] T. Safavi, J. Zhu, and D. Koutra, *NegatER: Unsupervised Discovery of Negatives in Commonsense Knowledge Bases*, 2021. DOI: 10.18653/v1/2021.emnlp-main.456.

[5] H. Arnaout, S. Razniewski, G. Weikum, and J. Z. Pan, *Negative knowledge for open-world wikidata*, 2021. DOI: 10.1145/3442442.3452339.

[6] E. T. Mueller, *Commonsense Reasoning*, Second Edition, S. Elliott, Ed. Elsevier, 2015, pp. 1–16, ISBN: 978-0-12-801416-5. DOI: 10.1016/b978-0-12-801416-5.00001-2.

[7] T. Chklovski, *Learner, A system for acquiring commonsense knowledge by analogy*, 2003. DOI: 10.1145/945645.945650.

[8] E. Davis and G. Marcus, "Commonsense reasoning and commonsense knowledge in artificial intelligence," *Commununications of the ACM*, vol. 58, pp. 92–103, 2015, ISSN: 0001-0782. DOI: 10.1145/2701413.

[9] N. Tandon, A. S. Varde, and G. de Melo, "Commonsense Knowledge in Machine Intelligence," *Sigmod Rec*, vol. 46, no. 4, pp. 49–52, 2018, ISSN: 0163-5808. DOI: 10.1145/3186549.3186562.

[10] A. Krebs and D. Paperno, *Capturing discriminative attributes in a distributional space: Task proposal*, 2016. DOI: 10.18653/v1/w16-2509.

[11] A. Balayn, G. He, A. Hu, J. Yang, and U. Gadiraju, "Ready Player One! Eliciting Diverse Knowledge Using A Configurable Game," in *Proceedings of the ACM Web Conference 2022*, ser. WWW '22, New York, NY, USA: Association for Computing Machinery, Apr. 25, 2022, pp. 1709–1719, ISBN: 9781450390965. DOI: 10.1145/3485447.3512241.

[12] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," vol. 25, pp. 161–197, 1998, ISSN: 0169-023X. DOI: 10.1016/s0169-023x(97)00056-6.

[13] L. v. Ahn, "Games with a Purpose," *Computer*, vol. 39, no. 6, pp. 92–94, 2006, ISSN: 0018-9162. DOI: 10.1109/MC.2006.196. [Online]. Available: https://doi.org/10.1109/MC.2006.196.

[14] A. Balayn, G. He, A. Hu, J. Yang, and U. Gadiraju, "FindItOut: A Multiplayer GWAP for Collecting Plural Knowledge," 2021.

[15] S. Klamt, U.-U. Haus, and F. Theis, "Hypergraphs and Cellular Networks," *PCBI*, vol. 5, e1000385, 2009, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000385.