Towards a GLUCOSE Knowledge Graph

and the Power of its Generalization

[GitLab]

Bachelor Thesis

19.04.2022

Nadia Arslan

nwarslan@cl.uni-heidelberg.de

Institut für Computerlinguistik

Ruprecht-Karls-Universität Heidelberg

Supervisor Prof. Dr. Anette Frank

Reviewers Prof. Dr. Anette Frank

Prof. Dr. Michael Herweg

Matrikelnummer: 2828861

Abstract

The GLUCOSE dataset [Mostafazadeh et al., 2020] is a collection of children's short stories that are annotated with inference rules to capture commonsense knowledge around these stories. A distinctive feature of the dataset is the semi-structured form of the annotations, as each rule is represented by a specific statement and a generalized form of that statement. Although models trained on GLUCOSE knowledge achieve high results matching humans' capacities, the question on how the generalizing factor contributes to that is not yet answered. Does the generalization of the inference rules lead to a denser representation of knowledge? Or are these structures already present in the data and can be detected by state-of-the-art methods?

In this thesis **"Towards a GLUCOSE Knowledge Graph - and the Power of its Generalization"**, we adress this question by deriving a specialized and a generalized knowledge graph for every story. To enable a comparison, the annotated rules of each story are parsed into an Abstract Meaning Representation [Banarescu et al., 2013] whose structure is that of a graph. To obtain a specific and a generalized graph for every story, we merged the AMR parses of the specific and the generalized rules respectively. On the specific side a state-of-the-art algorithm for Coreference Resolution was applied to find the concepts to be merged. On the generalized side these concepts are already highlighted by the way the data is annotated. With this approach, it is possible to measure the difference between the specific and the generalized knowledge in terms of graph density.

Our results show that identifying coreferent concepts automatically leads to the same gain of graph density as manual annotations do.

Zusammenfassung

Der GLUCOSE-Datensatz [Mostafazadeh et al., 2020] ist eine Sammlung von kurzen Kindergeschichten, die mit Inferenzregeln annotiert sind, um allgemeines Weltwissen über diese Geschichten zu erfassen. Ein besonderes Merkmal des Datensatzes ist die halbstrukturierte Form der Annotationen, da jede Regel durch eine spezifische Aussage und eine verallgemeinerte Form dieser Aussage dargestellt wird. Obwohl Modelle, die auf der Grundlage von GLUCOSE-Wissen trainiert wurden, hohe Ergebnisse erzielen, ist die Frage, inwieweit der Generalisierungsfaktor dazu beiträgt, noch nicht beantwortet. Führt die Verallgemeinerung der Inferenzregeln zu einer dichteren Repräsentation des Wissens? Oder sind diese Strukturen bereits in den Daten vorhanden und können durch State-of-the-Art-Methoden erkannt werden?

In dieser Arbeit **"Towards a GLUCOSE Knowledge Graph - and the Power of its Generalization"** gehen wir dieser Frage nach, indem wir für jede Geschichte einen spezifischen und einen verallgemeinerten Wissensgraphen ableiten und diese Graphen dann hinsichtlich ihrer Dichte vergleichen. Um einen Vergleich zu ermöglichen, werden die annotierten Regeln jeder Geschichte in eine "Abstract Meaning Representation" (AMR) [Banarescu et al., 2013] geparst, deren Struktur die eines Graphen ist. Um einen spezifischen und einen verallgemeinerten Graphen für jede Geschichte zu erhalten, haben wir die AMR-Parses der spezifischen bzw. verallgemeinerten Regeln zusammengeführt. Auf der spezifischen Seite wurde ein Algorithmus zur Koreferenzauflösung angewandt, um koreferente Konzepte zu finden. Auf der verallgemeinerten Seite sind diese Konzepte bereits durch die Art und Weise, wie die Daten annotiert sind, hervorgehoben. Mit diesem Ansatz ist es möglich, den Unterschied zwischen dem spezifischen und dem verallgemeinerten Wissen bezüglich der Graphendichte zu messen.

Unsere Ergebnisse zeigen, dass die automatische Identifizierung koreferenter Konzepte zu demselben Gewinn an Graphendichte führt wie manuelle Annotationen.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

In the present work, we want to discuss the gain of knowledge that results from including generalized rules in the annotations of GLUCOSE [Mostafazadeh et al., 2020]. To get a better insight into the structure of GLUCOSE knowledge, we describe the dataset in chapter 2 and give some examples of the way the stories are annotated. Chapter 3 discusses the methods of graph design on three levels: in section 3.1 we look at the data on story level and apply a clustering algorithm to classify the stories according to their content, in section 3.2 we describe how we cluster the annotations on sentence level, and in section 3.3 we illustrate how we parse these annotations into an AMR structure [Banarescu et al., 2013] and merge its coreferent concepts. We do this for the specific and the general annotations respectively and then compare the emerging graphs in chapter 4 and discuss the results. In chapter 5, we conclude that with our approach, we find no significant difference concerning the gain of knowledge between the specific and the general graphs.

2 GLUCOSE

GLUCOSE [Mostafazadeh et al., 2020] stands for "GeneraLized and COntextualized Story Explanations". It is a selected dataset based on the ROCStories corpus [Mostafazadeh et al., 2016] with a "focus on children's stories due to their simpler language and concepts" (4.2. p.4573). Given a story S and a sentence X in the story, 371 crowd workers were asked to annotate inference rules to capture implicit commonsense knowledge about the world. "Each entry includes a story-specific causal statement paired with an inference rule generalized from the statement"(Abstract p.4569). "To ensure diverse viewpoints and hypotheses, each S, X pair was assigned to three workers." (4.2. p. 4573)

The dataset contains 4584 stories, every story consists of five sentences. We excluded 297 from original 4881 stories with incorrect punctuation marks to maintain uniformity. To illustrate our work, we give an example story as reference for further explanations:

1. Tom loves the ocean.
2. He vacations to the beach often.
3. He decides to buy a vacation home.
4. He got one right on the beach.
5. Tom spent a lot of time there.

Each annotation has a specific and a general form, as the following three rule examples illustrate:

Tom love(s) the ocean	ENABLES	Tom vacations to the beach often	(a)
Someone_A love(s) the ocean	ENABLES	Someone_A vacations to the beach often	
Tom vacations to the beach often	ENABLES	Tom got a home on the beach	(b)
Someone_A vacations to Somewhere_A	ENABLES	Someone_A got a home Somewhere_A	
Tom possesses lots of money	ENABLES	Tom vacations to the beach often	(c)
Someone_A possesses lots of money	ENABLES	Someone_A vacations often	

The annotators were free to decide which concepts to generalize and how to realize that. There was only one constraint, namely that these - we call them generalizers - must be indicated by an underscore in the annotation. For instance, in our example rules the generalizer "Someone_A" does not only link the subjects of the consequent and the antecedent for rule (a), (b) and (c) respectively, it also indicates that the concept generalized is a person.

We extracted 155977 generalizers from the generalized rules and labeled them with one of five categories: PERSON, OBJECT, LOCATION, ANIMAL and OTHER. A complete list with information on how we categorized the generalizers can be found in our GitLab repository[1].

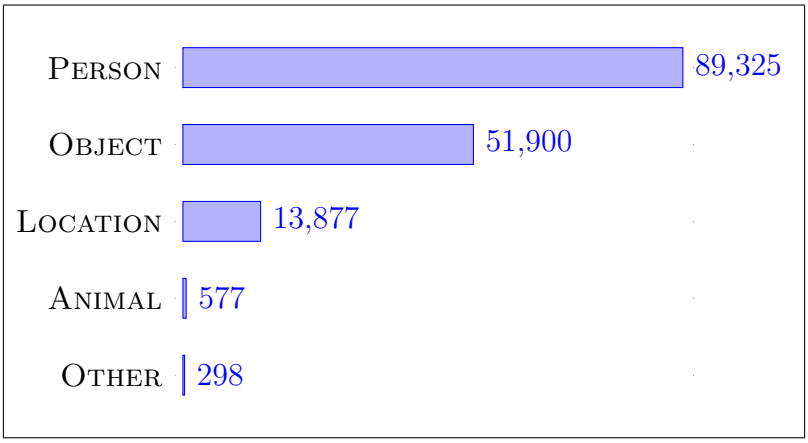

Figure 1: Distribution of Generalizers in GLUCOSE

Figure 1 shows the distribution of the labeled generalizers. More than half of the annotated generalizers are labeled with PERSON, approximately a third with OBJECT and less than a tenth with LOCATION. The categories ANIMAL and OTHERS occur rarely and account less than 1% of the generalizers.

1 https://gitlab.cl.uni-heidelberg.de/nwarslan/glucose_knowledge/generalizers

3 Graph Design

In this chapter, we describe the methods we use to build a GLUCOSE knowledge graph and explain the design choices we made. We look at the data on three levels and combine the structural information that we derive from (i) a story clustering (chapter 3.1), (ii) a sentence clustering (chapter 3.2) and (iii) an AMR parsing and merging (chapter 3.3) process.

A graph G is formally defined as a set of nodes N and a set of edges E connecting these nodes, so that $E \subseteq \{(n, m)|n, m \in N\}$. A graph can thereby represent relations (i.e. edges) between various components (nodes), which makes it the common structure to capture semantic meaning. Not only all the big knowledge resources today (e.g.: Wikipedia or Google), but also many of the semantic frameworks (e.g. AMR or RDF[1]), are represented in that structure.

AMR [Banarescu et al., 2013] stands for "Abstract Meaning Representation" and is a semantic representation language. An AMR graph is rooted and directed. A sentence parsed to an AMR graph is stripped down to its relevant concepts represented as nodes, and the relation between these concepts represented as edges[2]. The AMR structure is designed to abstract from the syntactic features of language and therefore information like numerus or tempus is not represented in an AMR graph.

To integrate AMRs in our knowledge graph we use AMRlib [Banarescu et al., 2013], a python module to parse sentences into an abstract meaning representation. The penman graph notation [Goodman, 2020] allows to modify and merge AMR graphs and we use it to rename node variables and restructure the AMR parses. To implement the final graph structure we use networkX [Hagberg et al., 2008], a python package to create and analyze graphs (i.e. networks). The package comes with a variety of inbuilt functions to visualize and evaluate networks.

1 RDF is a Resource Description Framework to uniform, map and interchange data on the web [Klyne, 2004]
2 The concepts and relations are coming from ProbBank [Palmer et al., 2005]

3.1 Story Level

To gain an oversight of the stories' topics, we apply a clustering algorithm. For this, we first embed every story with SentenceTransformers [Reimers and Gurevych, 2019], a Python framework for state-of-the-art sentence, text and image embeddings. To cluster these embeddings, we use the Python libary HDBSCAN [McInnes et al., 2017] (Hierarchical Density-Based Spatial Clustering of Applications with Noise). HDBSCAN decides by itself how many clusters it will form but takes a variable to set the minimum cluster size.

To get the clusters' topic words, we compute the most frequent words per cluster via TF-IDF score. The term frequency - inverse document frequency (TF-IDF) is a statistical measure to evaluate how relevant a word is to a document in a collection of documents. To apply this measurement to our task, we merge all the stories of one cluster to one document and then compute the TF-IDF score for every word in that document.

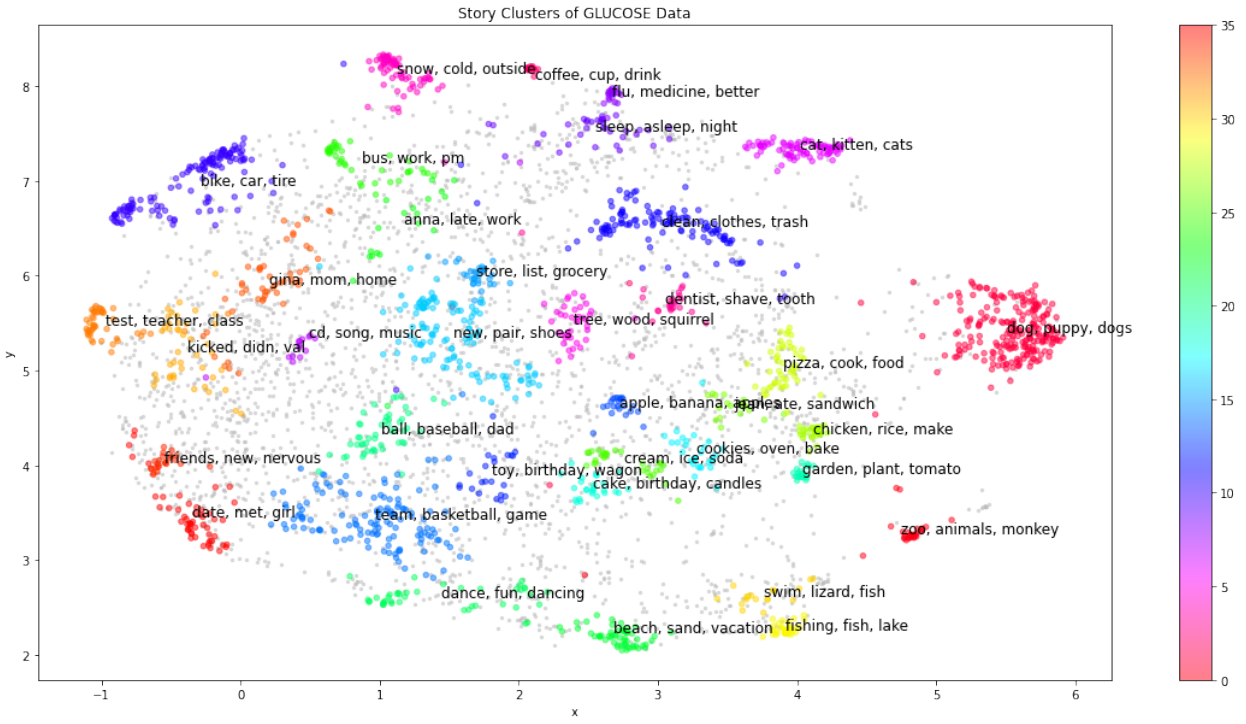

Figure 2: Story Clustering

Figure 2 shows the result[3] of the clustering algorithm with minimum cluster size set to fifteen.

3 A complete list can be found in our GitLab Repository in ./code/1_preprocessing.ipynb (chapter 3.2 out 13)

Every point stands for a story, every color[4] for a cluster, annotated with its top three most frequent words. To illustrate the result in a 2D coordinate system, we reduce the dimensionality of the embeddings with UMAP [McInnes et al., 2018]. Uniform Manifold Approximation and Projection (UMAP) is a dimension reduction technique that can be used to visualize data points of higher dimensionality.

AMR parsing is a resource-hungry and time consuming process. For this reason, we only parse stories form certain topics, following the clustering resu in figure 2. We list and group the hand-picked topics in table 1:

Group	ID	Top 3 Words	$\sum$ Stories	$\sum$ Rules
Animal	0	zoo, animal, monkey	34	2139
	1	dog, puppy, pet	231	15171
	6	cat, kitten, Sarah	84	5382
			349	22692
Sea	22	beach, sand, vacation	57	3794
	29	fishing, fish, lake	40	2802
	30	swim, lizard, fish	18	1138
			115	7734
Food	13	apple, bananas, tree	22	1538
	17	cookies, oven, bake	29	2049
	19	garden, plant, tomato	18	1180
	25	cream, ice, soda	31	2166
	27	chicken, rice, make	29	1877
	28	pizza, cock, food	48	3611
			177	12421
ALL	-	-	641	42847

Table 1: Selected Topics for AMR Parsing

The group of animal-related stories includes 22692 rules and 349 stories, the group of sea-related and food-related stories 7734 and 12421 rules respectively. In total, we selected 641 stories annotated with 42847 rules.

4 The gray points represent stories that were not assigned to a cluster

3.2 Sentence Level

To built a story graph based on the annotated sentences, we start creating a node for every sentence in the story. A rule is now represented as two nodes connected through an inference edge. Since every annotation is based on a specific sentence of its associated story, we can label one side of every rule (either the antecedent or the consequent) accordingly. For example, the first part of rule (a) refers to sentence 1 of our story example. The second part is the actual annotated part. In our example rule, this part refers to story sentence 2. Figure 3 shows the simple case were the first sentence entails the second:

Figure 3: Rule (a) Represented as Two Nodes Connected Through an Inference Edge

But this is not necessarily the case for other rules. The antecedent of rule (c), for example, is not a sentence of the story, but a sentence that the annotator inferred from his or her commonsense knowledge about the world while reading the story. We will call these sentences "external sentences" (e_n) and distinguish them thereby from the story's "sentence nodes" (s_n). To find out if a sentence should be represented by a sentence node or an external node, we compute the cosine similarity[5] between the sentence and the story's sentences. If the cosine similarity to the most similar story sentence is higher than 0.5 we associate the sentence with the sentence node. We cluster the rest of the sentences, i.e. the external sentences, by their cosine similarity ($>$0.6) and assign an external node for every cluster. For instance the two external sentences "Tom has a lot of money" and "Tom has money" are sorted in the same cluster and therefore share an external node.

Figure 4 shows the result for our story example. Sentences 1 to 5 are represented as blue rectangle nodes (s_1 - s_5), and the inferred new information, the external nodes, as yellow rectangle nodes (e_1 - e_6)[6]. Due to the way the dataset is designed, it is not possible for two external nodes to share an edge. An external node can only be connected to one (*e.g.* e_4) or several (*e.g.* e_1) sentence nodes.

5 To compute the cosine similarity between sentences we embed them with SentenceBert [Reimers and Gurevych, 2019]

6 Note that figure 4 only shows exemplary 6 of the 29 external nodes the story actually has

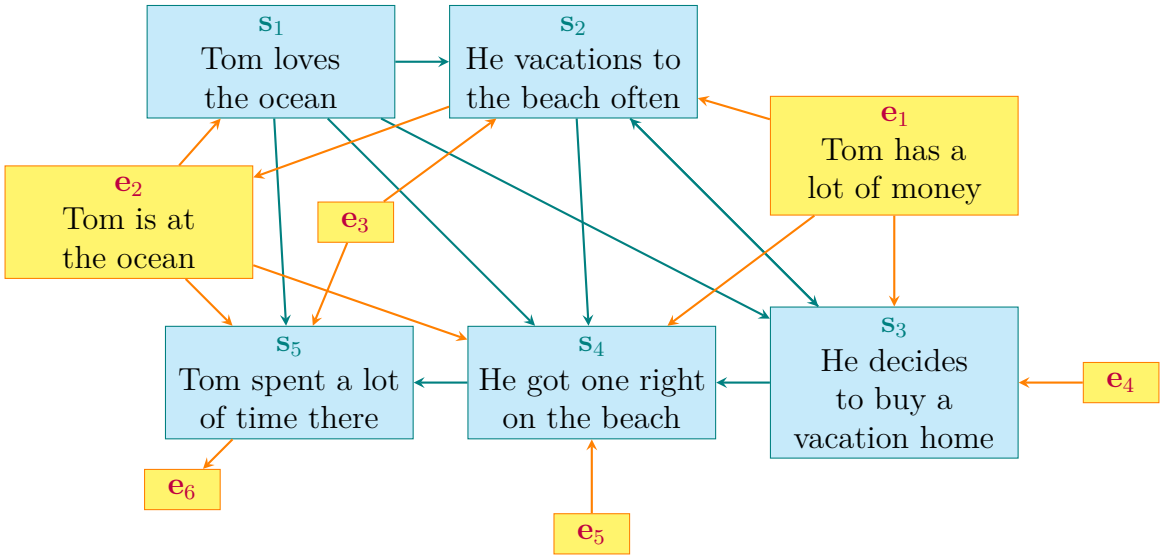

Figure 4: Tom Story Graph on Sentence Level

3.3 AMR Level

For every story of the selected topics in table 1, we want to build two merged AMR graphs, a specialized graph from the specialized rules (G_s) and a generalized graph from the generalized rules (G_g). In our graphs, we want coreferent concepts to share a node. In consequence, we have to merge certain nodes of the AMR parses. While keeping in mind that there are multiple annotations for a certain rule, we also want to extract coreference chains extending over multiple rules.

3.3.1 Merging AMRs of Multiple Annotations

In chapter 3.2, we cluster sentences with similar or identical meaning to build an inference graph on sentence level. We now use this graph to identify the annotations relating to the same or similar AMR parse. For example, for story sentence 1 there are eight different annotations and six partly distinct AMR parses (see figure 5). A feature of the Abstract Meaning Representation is that it does not capture the tense of verbs. Therefore, annotation 2.), 3.) and 4.) share the same AMR graph.

1.) Tom always loved the ocean

```
(l / love-01
    :ARG0 (p / person
        :name (n / name
            :op1 "Tom"))
    :ARG1 (o / ocean)
    :time (a / always))
```

2.) Tom loves the ocean

```
(l / love-01
    :ARG0 (p / person
        :name (n / name
            :op1 "Tom"))
    :ARG1 (o / ocean))
```

3.) Tom has love for the ocean

```
(l / love-01
    :ARG0 (p / person
        :name (n / name
            :op1 "Tom"))
    :ARG1 (o / ocean))
```

4.) Tom loved the ocean

```
(l / love-01
    :ARG0 (p / person
        :name (n / name
            :op1 "Tom"))
    :ARG1 (o / ocean))
```

5.) Tom loved the ocean and the beaches

```
(l / love-01
    :ARG0 (p / person
        :name (n / name
            :op1 "Tom"))
    :ARG1 (a / and
        :op1 (o / ocean)
        :op2 (b / beach)))
```

6.) Tom likes the beach

```
(l / like-01
    :ARG0 (p / person
        :name (n / name
            :op1 "Tom"))
    :ARG1 (b / beach))
```

7.) Tom likes the ocean and beaches

```
(l / like-01
    :ARG0 (p / person
        :name (n / name
            :op1 "Tom"))
    :ARG1 (a / and
        :op1 (o / ocean)
        :op2 (b / beach)))
```

8.) Tom likes the ocean

```
(l / like-01
    :ARG0 (p / person
        :name (n / name
            :op1 "Tom"))
    :ARG1 (o / ocean))
```

Figure 5: Different Annotations and AMR Parses of Story Sentence 1

To merge these graphs, we take the biggest graph as reference, assuming that it holds the most information. Iterating over the rest of the AMR graphs, we only add concepts and relations if we can't find them in the reference graph. The emerging graph in figure 6 has one node for every single concept but one concept can have the same relation to multiple other concepts. For instance the node "Tom" is in relation "ARG0" to the nodes "love" and "like", which in turn have the relation "ARG1" to several other concepts. The shades of color in figure 6 indicate the frequency of the concept in the annotations in figure 5. "Tom" can be found in every annotation and therefore has a darker shade than for example "always" which only occurs once. For every node in every story graph (see section 3.2), we merge the AMR parses in the way described.

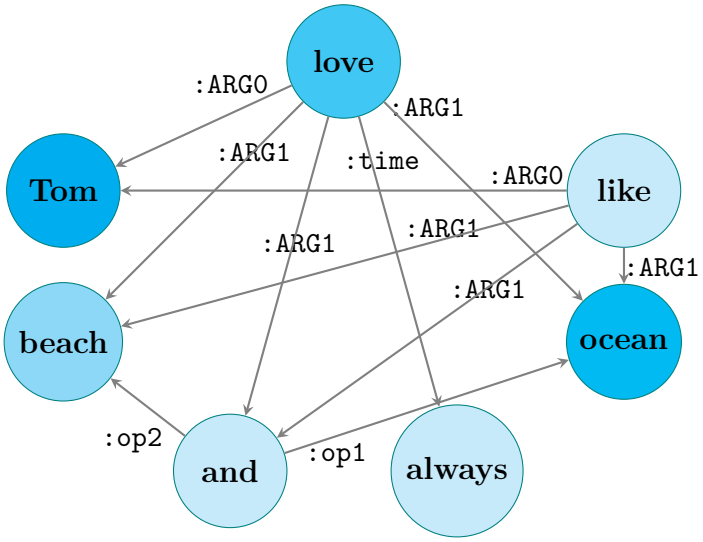

Figure 6: Merged AMR Graph of Story Sentence 1

3.3.2 Extracting Coreference Chains

To unite the merged AMR parses of every story to one merged graph, we derive coreference chains from the annotated rules. One chain should contain all triples that refer to the same concept. For example, from rule (a), (b) and (c) we want to derive that "Tom" or "Someone_A" is the subject (ARG0) of love, vacation, get and possess. But before we can concatenate coreferent triples, we have to extract them rule by rule. Algorithm 1 gives an overview of that process.

Algorithm 1 Extract Coreferent Triples

```
1: coref_triples = [ ]
2: for rule in story_graph do
3:     coreferences = resolve_coreference(rule)
4:     amrs = parse_to_amr(rule)
5:     triples = get_triples_of(coreferences, amrs)
6:     coref_triples.append(triples)
7: end for
8: return(coref_triples)
```

The input of the algorithm is a story graph with it's annotated rules (see section 3.2). For every rule, it resolves coreference (line 3), parses to AMR (line 4) and then extracts coreferent triples from the AMR parse (line 5). The output is a list of tuples where every tuple holds two such coreferent triples coming from the antecedent and the consequent of a rule respectively. In the following we see three examples of such tuples:

$$\text{example rule } (a) \rightarrow \quad \big((\text{love-01}, \texttt{:ARG0}, s_1), (\text{vacation-01}, \texttt{:ARG0}, s_2)\big) \tag{1}$$

$$\text{example rule } (b) \rightarrow \quad \big((\text{vacation-01}, \texttt{:ARG0}, s_2), (\text{buy-01}, \texttt{:ARG0}, s_3)\big) \tag{2}$$

$$\text{example rule } (c) \rightarrow \quad \big((\text{possess-01}, \texttt{:ARG0}, e_1), (\text{vacation-01}, \texttt{:ARG0}, s_2)\big) \tag{3}$$

Each triple consists of the concept[7] that takes the coreferent as argument, the relation the coreferent has to the concept and the sentence index the triple was extracted from. The triple "(vacation-01, :ARG0, s_2)" takes part in all three tuples. From this we can conclude that all three tuples refer to the same concept and therefore belong to the same coreference chain:

(love-01, :ARG0, s_1) – (vacation-01, :ARG0, s_2) – (buy-01, :ARG0, s_3) – (possess-01, :ARG1, e_1)

7 The attached number ending some of the concepts point to the PropBank word sense.

Table 2 shows some selected coreference chains for our story example. The first column lists all AMR triples from sentences $s_1 - s_5$ referring to "Tom". The second column of the table includes triples mentioning "ocean". The ocean is loved, the ocean is located by or near to something and also Tom is located by the ocean. The last column shows triples relating to "home". The home is bought, the home is sold, the home is possessed and the home is the location of spending (time).

Tom	**ocean**	**home**
(love-01, `:ARG0`, s_1)	(love-01, `:ARG1`, s_1)	(buy-01, `:ARG1`, s_3)
(like-01, `:ARG0`, s_1)	(by, `:op1`, s_3)	(sell-01, `:ARG1`, e_3)
(vacation-01, `:ARG0`, s_2)	(near-02, `:ARG2`, s_3)	(possess-01, `:ARG1`, e_1)
(decide-01, `:ARG0`, s_3)	(visit-01, `:ARG1`, e_4)	(spend-02, `:location`, s_5)
(buy-01, `:ARG0`, s_3)	(Tom, `:location`, e_2)	
(get-01, `:ARG0`, s_4)		
(spend-02, `:ARG0`, s_5)		

Table 2: Coreference Chains of Story Example

We extract these coreference chains for every story twice, based on the specific and the general rules respectively. To resolve the coreferences of the specific rules, we apply Huggingface's NeuralCoref[8] system to every rule. The coreference in the general rules is already resolved by the annotators.

3.3.3 Final Merge

To built a merged story graph, we combine the merged AMR sentence parses (section 3.3.1) and the extracted argument chains (section 3.3.2). For each coreference chain, we merge the corresponding nodes of the AMR sentences parses. Figure 7 illustrates this process for the first three sentences of our story example. In the upper part of the figure, we see the merged sentence parses of sentence 1 (blue), sentence 2 (green), and sentence 3 (orange). We only annotated relations referenced in the argument chains in the middle of the figure. The red merged nodes of the graph in the lower part of the figure represent the coreferent concepts derived from the argument chains.

In this way we build a specialized graph and a generalized graph for every story.

8 https://github.com/huggingface/neuralcoref

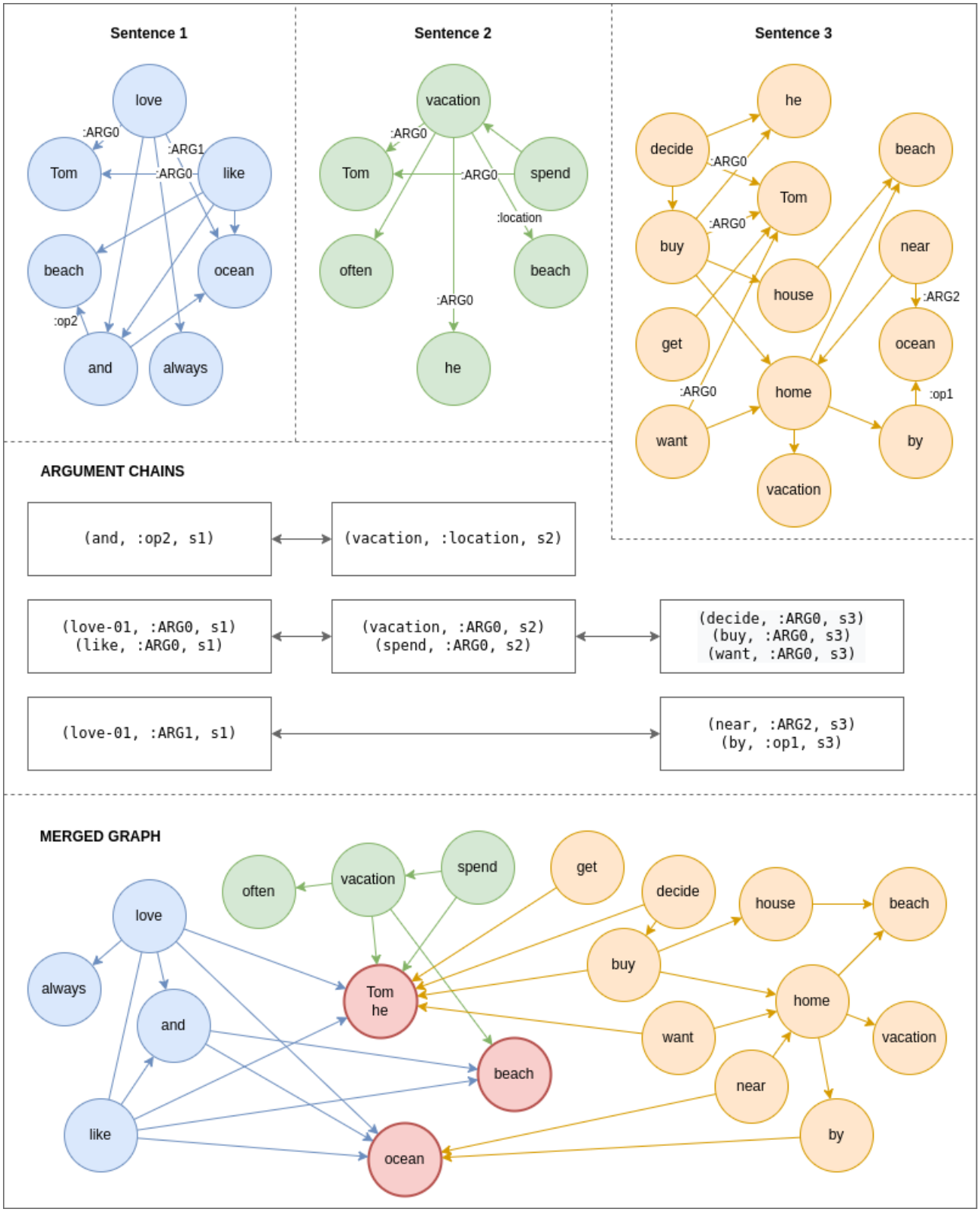

Figure 7: Merged Story Graph

4 Experiments and Results

To test the power of generalization in GLUCOSE, we compare the specialized graph against the generalized graph before and after the merge. We examine different aspects such as the number of nodes and edges, the average degree, and the density of the graphs.

Before the merge (in section 3.3), the number of nodes corresponds to the number of instances over all AMR parses of every annotated sentence and the number of edges corresponds to the number of relations in these parses. After the merge, we expect that the number of nodes and edges decreases while the value of density increases.

The density D of a graph is formally described as the ratio between the number of actual edges E_{actual} and the number of all possible edges $E_{possible}$:

$$D = \frac{E_{actual}}{E_{possible}} \qquad \textit{with} \quad E_{actual} = |E| \quad \textit{and} \quad E_{possible} = \frac{|N| \cdot (|N| - 1)}{2}$$

D scales from 0 to 1 where 0 indicates a graph without edges and 1 a fully connected graph, i.e. every node is connected to every other node. We compute the density for undirected graphs, although the graphs we built are directed. We argue that in an AMR graph, a relation between two nodes can only go in one direction, not both. Therefore, $E_{possible}$ has the value of an undirected graph[1]. Reducing the numbers of nodes of a graph will lower the value of $E_{possible}$ and therefore lead to an increase of density. That is the reason we expect the merged graph to be denser than the unmerged graph.

The degree distribution of a graph tells us how many nodes hold a certain degree of edges. The more nodes with high degree, the better connected the graph structure. The average degree of a graph is defined as the ratio between the sum of every node's degree and the number of nodes:

$$\varnothing_{degree} = \frac{\sum Degree(N_i)}{|N|}$$

1 In a directed graph $E_{possible} = |N| \cdot (|N| - 1)$

Table 3 shows the number of nodes and edges, the density, the average degree and the number of coreference chains (N_{coref}) for our Tom story and for all 641 stories on average. For both the specific and the general graph, we list the results before and after the merge and the difference between the two.

		Tom Story					All Stories ($\sum$641)				
		nodes	edges	density	$\varnothing_{degree}$	N_{coref}	nodes	edges	density	$\varnothing_{degree}$	N_{coref}
Specific	unmerged	397	348	0.004	1.75	5	400	358	0.005	1.79	2
	merged	131	186	0.022	2.84		167	192	0.016	2.29	
	difference	-266	-162	+0.018	+1.09		-233	-166	+0.011	+0.5	
General	unmerged	451	391	0.004	1.73	3	362	302	0.005	1.67	3.5
	merged	163	222	0.017	2.72		170	199	0.016	2.33	
	difference	-288	-169	+0.013	+0.99		-192	-103	+0.011	+0.66	

Table 3: Comparison of Specific and General Graph Before and After Merge

The results show that with merging coreferent concepts, we reduce the number of nodes and edges and increase the value of density. The increase of density of our Tom story is higher for the specialized graph than for the generalized. But if we consider the increase of density in all stories on average, this difference disappears. The results also show that with merging, we increase the average degree of the graphs but a significant difference between the specific and the general graph does not emerge. On average we find 2 coreferent chains on the specific side and 3.5 on the general side. We only mention these numbers to provide background knowledge, since we cannot evaluate how many chains are actually in each story without counting them manually. For our Tom story we can say that we found 3 coreference chains on the general side referring to (i) Tom, (ii) home and house, and (iii) beach and ocean. On the specific side we found 5 chains referring to (i) Tom, (ii) ocean, (iii) home and house, (iv) beach, and (v) beach, which is not as correct as the general chains.

		Animal Stories ($\sum$349)			Sea Stories ($\sum$115)			Food Stories ($\sum$177)		
		nodes	edges	density	nodes	edges	density	nodes	edges	density
Specific	unmerged	398	358	0.005	400	362	0.005	406	356	0.005
	merged	164	189	0.016	173	199	0.015	169	193	0.015
	difference	-234	-169	+0.011	-227	-163	+0.010	-237	-163	+0.010
General	unmerged	361	302	0.005	373	316	0.005	356	292	0.005
	merged	170	198	0.016	181	212	0.015	165	193	0.016
	difference	-191	-104	+0.011	-192	-104	+0.010	-191	-99	+0.011

Table 4: Results of Animal, Sea and Food Stories Before and After Merge

Table 4 compares the density of animal, sea and food stories. In the largest subset, the group of animal stories, we see the gain of density equal to the set of all stories. In the category food, the general graph is slightly denser than the specific one.

5 Conclusion

With our approach we show that with merging coreferent concepts of AMR parses, we increase the density of their associated graphs. We also show that including manually annotated rules that generalize from a statement and resolve coreferences does not lead to a denser representation of knowledge. There are state-of-the-art algorithms that come to the same result. However, which method resolves coreference more accurate is another question that we cannot answer in this work.

Bibliography

Nasrin Mostafazadeh, Aditya Kalyanpur, Lori Moon, David Buchanan, Lauren Berkowitz, Or Biran, and Jennifer Chu-Carroll. GLUCOSE: GeneraLized and COntextualized Story Explanations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4569–4586, 2020.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `https://aclanthology.org/W13-2322`.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, 2016.

Graham Klyne. Resource Description Framework (RDF): Concepts and Abstract Syntax. 2004. URL `www.w3.org/TR/rdf-concepts`.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106, 03 2005. ISSN 0891-2017. URL `https://doi.org/10.1162/0891201053630264`.

Michael Wayne Goodman. Penman: An Open-Source Library and Tool for AMR Graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 312–319, Online, July 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.acl-demos.35`.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language*

Processing. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical Density based Clustering. *The Journal of Open Source Software*, 2(11):205, 2017.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. UMAP: Uniform Manifold Approximation and Projection. *The Journal of Open Source Software*, 3(29):861, 2018.