Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 3984–3993
Marseille, 20-25 June 2022

Towards the Detection of a Semantic Gap in the Chain of Commonsense Knowledge Triples

Yoshihiko Hayashi
Faculty of Science and Engineering, Waseda University
3-14-9 Ohkubo, Shinjuku, Tokyo 1690072, Japan
yshk.hayashi@aoni.waseda.jp

Abstract

A commonsense knowledge resource organizes common sense that is not necessarily correct all the time, but most people are expected to know or believe. Such knowledge resources have recently been actively built and utilized in artificial intelligence, particularly natural language processing. In this paper, we discuss an important but not significantly discussed the issue of semantic gaps potentially existing in a commonsense knowledge graph and propose a machine learning-based approach to detect a semantic gap that may inhibit the proper chaining of knowledge triples. In order to establish this line of research, we created a pilot dataset from ConceptNet, in which chains consisting of two adjacent triples are sampled, and the validity of each chain is human-annotated. We also devised a few baseline methods for detecting the semantic gaps and compared them in small-scale experiments. Although the experimental results suggest that the detection of semantic gaps may not be a trivial task, we achieved several insights to further push this research direction, including the potential efficacy of sense embeddings and contextualized word representations enabled by a pre-trained language model.

Keywords: commonsense knowledge graph, word-sense disambiguation, sense embeddings, ConceptNet.

1. Introduction

Commonsense knowledge is fundamental and mostly implicit knowledge employed by humans to live reasonable and safe lives. Therefore, the acquisition, representation, and exploitation of commonsense knowledge have long been a central issue in artificial intelligence that aims to realize human-level smartness in several application domains (Davis and Marcus, 2015). In particular, commonsense knowledge resources have been widely used in natural language processing (NLP) to "read between the lines" (Rashkin et al., 2018), which could contribute to solving a given task, such as question-answering (Talmor et al., 2019; Feng et al., 2020).

AAmong the several types of commonsense knowledge resources (Ilievski et al., 2021), ConceptNet (Speer et al., 2017) may be one of the most popular resources, as it maintains a large number of knowledge assertions in multiple languages. Most of the work using this resource treats it as a gigantic graph and tries to benefit from a path that connects a starting node with an ending node (Lin et al., 2019; Paul and Frank, 2019). For example, the path ⟨computer →UsedFor →work →AtLocation^{-1} →office⟩ in the graph shown in Figure 1 hints the relation between *computer* and *office* via *work*. On the other hand, adoption of the path ⟨edge_tool →IsA^{-1} →plane →UsedFor →travel⟩ may cause a problem because usually an edge tool has nothing to do with travelling. This issue is caused by the polysemous nature of the node labeled "plane."

In the present work, we refer to the issue exemplified by such a path as **semantic gap** issue and try to establish a method to detect potential semantic gaps in a path in advance to the exploitation in a commonsense reasoning process. As our approach relies on machine learning, we created a pilot dataset using ConceptNet and trained a few baseline classifiers to detect a semantic gap in the path of length two. We name such a path as **knowledge triple chain**, which is considered a fundamental building block to form a longer path. A set of pilot experiments were then conducted to explore potentially effective features in the classification.

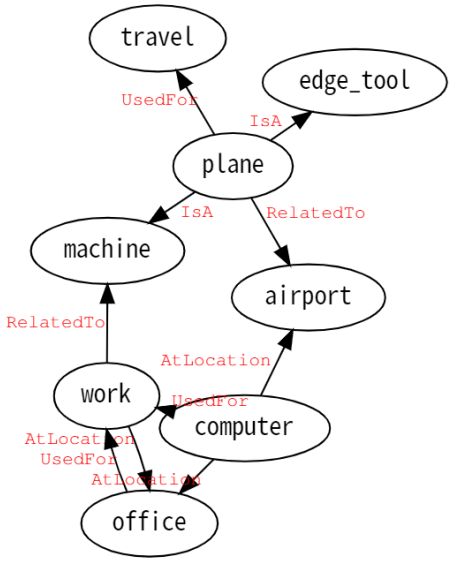

Figure 1: A part of ConceptNet. Each node is associated with a concept, or more precisely, an implicit set of concepts, associated with a word or phrase that is used as the node label. A directed edge represents an inter-node relation labeled with the relation type.

Our contributions in this paper are summarized as follows.

- Although not highly formal, we defined the issue of semantic gap in commonsense knowledge chain and argued that the approach with precise word sense disambiguation should not be feasible nor desirable.
- We created an initial dataset of knowledge triple chains[1] using ConceptNet, each labeled whether it involves a semantic gap or not.
- We trained a few baseline classifiers and compared them in pilot experiments, confirming that sense embeddings ARES (Scarlini et al., 2020) and contextualized word representations achieved by BERT (Devlin et al., 2019) could be effectively employed in detecting semantic gaps.

2. Semantic Gaps in a Commonsense Knowledge Grpah

In the present work, we work on a commonsense knowledge resource known as ConceptNet[2] (Speer et al., 2017). ConceptNet has been widely used in research projects that deal with commonsense, as it is a large multilingual resource that is freely available. It, however, poses several issues that should be addressed. Most of these issues stem from the innate nature of commonsense as well as the resource construction process.

2.1. ConceptNet and the Semantic Gap Issue

ConceptNet maintains a large number of triples, each representing a fragmental or episodic commonsense knowledge. Many of these triples have been collected by the Open Mind Common Sense[3] crowdsourcing project. These triples are generally considered as collectively forming a large graph.

However, as pointed out by Zhou et al. (2019), ConceptNet has several issues, such as unverified noisy triples, under-specified triples that are only relevant in a specific context/situation, and the nodes that are generally not being disambiguated. In the present work, we focus on potential semantic gaps that could be found in an arbitrarily chosen path in the ConceptNet graph. This issue can be primarily attributed to the nodes overloaded by multiple meanings.

Figure 1 is presented to exemplify the issue of semantic gaps. The figure depicts a part of ConceptNet represented as a graph, where we see four triples around the node labeled "plane." Three of them are associated with the *airplane* sense of "plane," while one of them is related to the *tool* sense of "plane." This means that we may be in danger of taking an inadequate path, such as ⟨edge_tool →$\texttt{IsA}^{-1}$ →plane →$\texttt{UsedFor}$ →travel⟩, in conducting a sort of commonsense reasoning. Obviously, an edge tool has nothing to do with traveling.

This **semantic gap issue** arises whenever a common node is polysemous, and different meanings are intended in the adjoined triples. An ontologically correct way to resolve this issue is segregating a polysemous node according to the intended meanings and rearranging the corresponding edges. This reconstruction process manifestly requires precise word-sense disambiguation. This solution may be not only costly but could affect the usability of the commonsense knowledge resource. Thus, we would like to establish a way to only adopt a path without or less explicit semantic gaps by making use of machine-learning technologies.

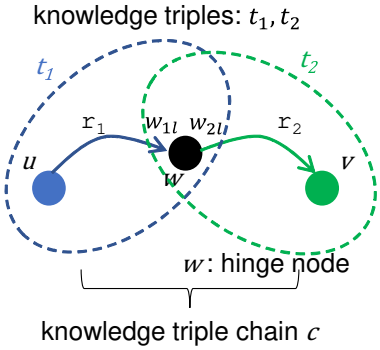

Figure 2: A knowledge triple chain c consists of two adjacent knowledge triples, t_1 and t_2. The directionality of an edge is arbitrary, as we can think of the inversely directed relation (e.g., $\texttt{IsA}^{-1}$) given a directed relation (e.g., $\texttt{IsA}$). Also, an undirected relation, such as $\texttt{SimilarTo}$ can be represented by a set of two oppositely directed triples.

2.2. Definitions and Notations

A **knowledge triple** $\langle x, r, y\rangle$ consists of two concept nodes, x and y, and the edge in between with the relation label r. Knowledge triples in ConceptNet are referred to as assertions[4]. The inventory of the ConceptNet relations is presented on the Web page[5], in which verbal definitions of 34 relation types are given.

As illustrated in Figure 2, a **knowledge triple chain** c, in the present work, is formed by two adjacent knowledge triples: $t_1 = \langle u, r_1, w\rangle$ and $t_2 = \langle w, r_2, v\rangle$, in which the node w is shared by t_1 and t_2. In the following, the node w is referred to as **hinge node**, and the corresponding word used as the label as **hinge word**, denoted as w_l. Notice that a knowledge triple chain can be considered as a primary component of an arbitrary

[1] https://sites.google.com/site/yoshihikohayashi/semantic-gaps-in-conceptnet-triple-chains

[2] https://conceptnet.io/

[3] https://media.mit.edu/projects/open-mind-common-sense

[4] https://github.com/commonsense/conceptnet5/wiki/Downloads

[5] https://github.com/commonsense/conceptnet5/wiki/Relations

length **knowledge path** p.

We say that a **semantic gap** exists in the knowledge triple chain c if the intended meaning of w with respect to t_1 may be different from the one with respect to t_2. Note that this is not a very rigorous definition but allows us to benefit from the commonsense knowledge resource that maintains not necessarily rigorously correct in the ontological sense but potentially useful knowledge compositions.

3. Creation of the Pilot Dataset

We started with creating a pilot dataset in which human assessments on semantic gaps are collected. With this dataset, we can devise and compare machine learning-based methods for semantic gap detection.

3.1. Selection of Knowledge Triple Chains

To initiate the dataset creation process, knowledge triple chains have to be sampled from ConceptNet. We first selected hinge nodes whose label words are polysemous but not highly abstract. We initially selected the words from the "List of English homographs"[6], and added some words which are largely known as polysemous. We subsequently consulted the ConceptNet (version 5.7.0) assertions and sampled knowledge triple chains around these candidate hinge nodes, which amount to 4,316 chains. In the sampling process, we excluded knowledge triples whose relation is highly vague or lexical[7].

3.2. Labeling of Knowledge Triple Chains

In this step, each knowledge triple chain is judged either of *without-gap* or *with-gap* and labeled accordingly. For example, the chain ⟨edge_tool $\rightarrow$`IsA`$^{-1}$ $\rightarrow$plane $\rightarrow$`UsedFor` $\rightarrow$travel⟩, is labeled *with-gap*, while ⟨machine $\rightarrow$`IsA`$^{-1}$ $\rightarrow$plane $\rightarrow$`UsedFor` $\rightarrow$travel⟩ is labeled *without-gap*, respectively.

Among the sampled chains, 3,000 of them were randomly chosen and fed into the human labeling process. An English native speaker did the first-round labeling, and then the author of this paper adjusted the initial decisions. In the labeling process, the annotators were allowed to consult WordNet (Fellbaum, 1998) synsets with their definitions, as well as the super-senses (lexicographer file names). In general, if each of the meanings of a hinge node word is likely to have different super-senses, the node is judged to have a semantic gap. As the given context is limited, judging if the given knowledge triple chain has a semantic gap in the hinge concept was sometimes quite hard even for the English native speaker. Besides, during the labeling process, we profoundly recognized that ConceptNet includes a certain amount of invalid or spurious knowledge triples, as noted by (Zhou et al., 2019). We flagged these knowledge triple chains as erroneous. However, for the sake of convenience, the knowledge triple chains containing these invalid triples were labeled as *with-gap*. This means that a knowledge triple chain labeled *with-gap* should not be used in reasoning, while that labeled *without-gap* may be safe to use.

To help the labeling process, we applied a set of relation-specific checking rules to mechanically detect presumably invalid knowledge triples (Appendix-A). Each of the rules looks at the potentially valid set of parts-of-speech for the hinge node word. For examples: `Synonym` relation requires the POS of w in t_1 and that in t_2 is identical; `CapableOf` specifies the POS of w in t_1 is noun while that in t_2 to be verb.

Here, we especially note that we finally decided to assign *without-gap* to a hinge node that carries mutually associated derivative meanings (e.g., "red" as a noun on one side and an adjective on the other side). In addition, we admitted a hinge node that exhibits so-called systematic polysemy (Peters and Peters, 2000) (e.g., "school" as *building* and *institution*) as *without-gap*. These decisions may not closely adhere to ontological principles. Nevertheless, we made these decisions expecting that they would allow more useful knowledge triple chains to be involved in the reasoning that uses knowledge paths.

3.3. Resulting Dataset

The resulting dataset maintains 3,000 unique knowledge triple chains, each carrying a semantic gap label, as well as a flag for indicating an invalid knowledge triple. Table 1 presents basic statistics of the resulting dataset. It shows that more than half of the chains are labeled *with-gap*, suggesting that an arbitrarily selected ConceptNet knowledge path may be dangerous to follow. Remind that a knowledge triple chain including at least one invalid triple is considered *with-gap* in the present work.

Item	Count
total # of chains	3,000
# of chains *without-gap*	1,313
# of chains *with-gap*	1,687
of which, # of invalid triples	262
# of unique triples	4,316
of which, # of triples flagged invalid	196
# of unique hinge words	255
# average degree of polysemy	10.5

Table 1: Statistics of the resulting dataset. The average degree of polysemy measures the average number of synsets classified in WordNet 3.0.

[6] `https://en.wikipedia.org/wiki/List_of_English_homographs`

[7] These relations are: `RelatedTo`, `HasContext`, `Antonym`, `DistinctFrom`, `DerivedFrom`, `EtymologicallyRelatedTo`, `EtymologicallyDerivedFrom`, and `FormOf`.

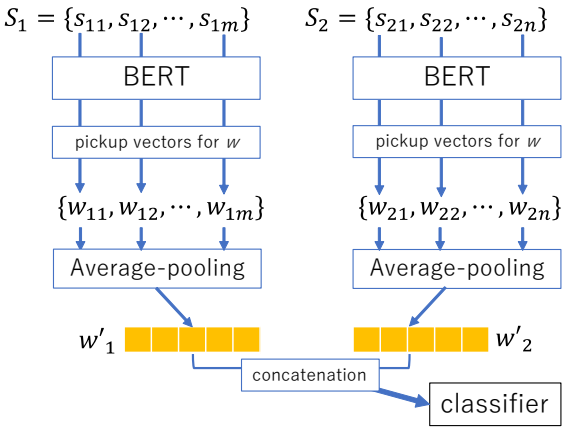

Figure 3: BERT-based baseline.

4. Baseline Detection Methods

This section presents baseline methods for detecting a semantic gap in the knowledge triple chain. These baselines and their combinations are compared in the preliminary experiments described in the next section. In the following, we often need to separate the representation of a hinge word w_l depending on the context given by each of the adjoining triples, t_1 and t_2. We respectively denote them as w_{l1} and w_{l2}.

4.1. BERT-based Baseline

Detecting a semantic gap in the given knowledge triple chain could be primarily accomplished by comparing the semantic nature of the hinge word w_l in t_1 context and that in t_2 context. That is, if these two are vastly different, we can expect that there is a semantic gap at the hinge node. Suppose we could capture appropriate context-dependent vector representations for w_{l1} and w_{l2}. In that case, these vectors can be used as features for the classifier in charge of detecting a semantic gap. By feeding a typical sentence that verbally represents the content of a knowledge triple into a sentence encoder such as BERT, we can extract the contextualized representation of the hinge word w_l in the triple context. We should, however, use multiple sentences with a variety, not a single sentence, in this process to improve the generality of the representation.

Figure 3 illustrates this approach. Given a knowledge triple chain, we generate the sets of sentences S_1 and S_2, respectively, for $t1$ and $t2$ by applying a set of relation-specific sentence generation templates, which we hand-coded. For example, sentences such as "A plane is used for a travel" and "You can use a plane to travel" are generated for the ConceptNet triple, ⟨plane, `UsedFor`, travel⟩. The average number of templates across the relation types is 3.67, with the maximum number being 14 for the `AtLocation` relation.

These generated sentences are fed into a BERT encoder, and we create a unified hinge word representation by average-pooling the contextualized vectors assigned to the hinge word in these generated sentences. The resulting vectors w'_1 and w'_2 are then concatenated and finally sent to the binary classifier. In the experiments, we primarily used a publicly available pretrained BERT encoder[8] and subsequently fine-tuned it with the present classification task.

4.2. ARES-based Baseline

A semantic resource such as WordNet can be consulted as a standard for semantic comparison. More specifically, the present work employs the WordNet senses/synsets of a hinge word w_l as the semantic foundation on which w_{l1} and w_{l2} are compared. By employing WordNet as a firm backbone, we expect that more robust features for semantic gap detection can be obtained than directly comparing the contextualized word representations.

Figure 4 overviews this approach, which relies on the WordNet sense inventory and the corresponding sense embeddings ARES (Scarlini et al., 2020). Note that the ARES embedding vectors are generated by making use of the same BERT encoder described in the previous subsection, which allows us to directly compare a word with a WordNet sense/synset. This is the exact reason for us to utilize the ARES embeddings.

Given a target hinge word w, we first consult WordNet and retrieve the corresponding senses θ_1 through θ_n, where n counts the number of senses of w. In parallel, the contextualized word representations w'_1 and w'_2 are generated by the same procedure described in the previous subsection. Subsequently, w'_1 and w'_2 are independently compared with each of the ARES sense vectors, yielding similarity vectors sv_1 and sv_2, respectively. The resulting vector sv_i captures the distribution of similarities of w'_i across the associated WordNet senses.

The size of the resulting similarity vectors sv_1 and sv_2 is determined by the number of senses of the hinge word w_l, meaning that it is not a constant across hinge words. Thus, the concatenation of sv_1 and sv_2 cannot be used as features on its own. Therefore, we summarize these vectors into an array of metrics for measuring the closeness of the distributions. We namely computed the Euclid distance, cosine similarity, KL-divergence, and correlation coefficients (Pearson's and Spearman's) between the similarity vectors. The resulting array of coefficients is finally fed into the binary classifier.

As discussed earlier in this paper, we believe that a soft or relaxed semantic matching approach should be preferred over a rigid sense-disambiguation approach. The use of semantic distribution can be a way to facilitate this direction. Besides, this approach could remedy the issue of too fine-grained sense distinctions made in WordNet.

[8]We used the BERT-large-uncased model provided by the HuggingFace transformers library that is available at `https://huggingface.co/docs/transformers`.

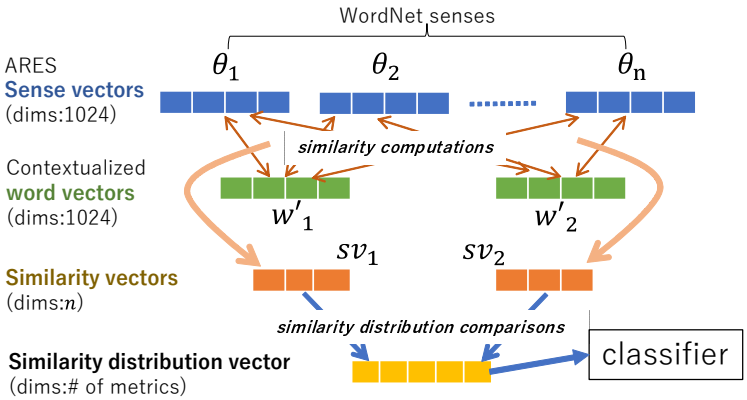

Figure 4: ARES-based baseline.

4.3. NumberBatch-based Baseline

Two words can be, to some extent, mutually disambiguated by seeking a sense combination that maximizes the sense-level similarity, provided a sense inventory and an effective similarity measure are available (Resnik, 1995). Following this classical idea, we estimate the similarity between two ConceptNet nodes, or more precisely, between their intended meanings in the knowledge triples. The similarities among the nodes that are computed in this way can be used as the features for detecting a semantic gap. More specifically, in the present work, we use similarities between ConceptNet nodes u and w_{1l}, w_{2l} and v, as well as that between u and v (refer to Figure 1), which are computed as described below. We explore the effectiveness of these similarities as features. They could approximate the validity of a knowledge triple chain involving particular combinations of relations, such as a consecutive occurrence of `SimilarTo` relations.

We estimate the similarity $sim_{CN}(x, y)$ between ConceptNet nodes x and y as formulated in (1): We adopt the maximum similarity between a possible combination of their WordNet synsets.

$$sim_{CN}(x, y) = \max_{a \in synsets(x), b \in synsets(y)} sim_{WN}(a, b). \quad (1)$$

The similarity $sim_{WN}(a, b)$ between two WordNet synsets a and b is then computed as follows[9]: We compute the averaged similarity between the pairs of the related synsets given by $rels(a)$ and $rels(b)$.

$$sim_{WN}(a, b) = \text{average}_{p \in rels(a), q \in rels(b)} sim_{NB}(p', q'). \quad (2)$$

In this specific computation, we use the NumberBatch word similarity $sim_{NB}(p', q')$ as the proxy of synset similarity. That is, p' and q' that respectively denote the first lemmas of the synsets are compared by using the NumberBatch word embeddings. The set of related synsets $rel(a)$ is created by consulting the WordNet graph: the k-neighbor synsets of a are first collected; then, presumably irrelevant synsets are excluded. In this filtering process, we measure the similarities between synsets by using the same NumberBatch-based similarity sim_{NB} and applying a threshold θ. In the experiments, we used the parameters $k = 2$ and $\theta = 0.8$. To sum up this procedure, we fundamentally adopt the classic idea of word sense disambiguation, but we exploit the graph structure of WordNet to enrich potentially useful information for the disambiguation. Besides, we exploit the NumberBatch word similarity as a proxy of sense-level similarity.

4.4. Additional Features: Relations and the Directionalities

Further features attainable from a knowledge triple chain are the involved relations and their directionalities. As these features are not informative enough on their own, we use these features as auxiliary features. Specifically, we vectorize these features by employing one-hot encoding and concatenate them with the primary feature vectors.

4.5. Alternative Baseline: Simple BERT-based Classifier

The contextualized word representations for w_{l1} and w_{l2} used in the BERT and ARES baselines could potentially be improved by fine-tuning the BERT encoder with the present target task. If this is the case, we could alternatively use the vectors obtained with the fine-trained BERT encoder.

To investigate this idea in the experiments, we fine-tune the BERT encoder by using the same set of sentences generated by the relation-specific templates. More specifically, an input data instance is formed as: [CLS] s_{1_i} [SEP] s_{2_j}. Here, s_{1_i} denotes the i-th sentence generated for t_1, and s_{2_j} designates the j-th sentence

[9] Instead of using the ARES embeddings, we explore the effectiveness of the NumberBatch embeddings with this baseline method.

generated for t_2, respectively. We distribute the gold semantic-gap label initially assigned to a knowledge triple chain to the corresponding sentence pairs.
In the experiments, we fine-tuned the BERT encoder with a classification head provided by the underlying training framework[10]. We used the AdamW optimizer with the learning rate 2e-5. The training with batch size eight was nicely converged with two epochs of iterations.

5. Experiments and the Results

5.1. Experimental Setup

The baselines described in the previous section and their combinations were compared in the preliminary experiments. In combining more than two baseline models, we simply concatenated their feature vectors. We used Logistic Regression, Random Forest, and Support Vector Machine as the binary classifier[11] and compared the performances. We only present the results with the Random Forest classifier in the following as it generally achieved better results than others. Detailed performance figures are given in Appendix-B.
As the currently available dataset is not large in size, we conducted four-fold cross-validation in the experiments: in each of the four runs, 75% of the data was used for training, and the rest 25% was used for testing. Needless to say, the same data splits were employed in fine-tuning the BERT encoder. We use macro-averaged precision, recall, and F1 to summarize the experimental results.

5.2. Results with Pre-trained BERT

Table 2 presents the results of baselines and their combinations. Remind that the BERT and ARES baselines employ BERT-originated contextualized word representations, while the NumberBatch baseline has nothing to do with these BERT vectors.

BERT	ARES	NumBat	P	R	F1
✓			0.68	0.65	0.65
	✓		0.64	0.61	0.60
		✓	0.66	0.65	0.65
✓	✓		0.69	0.65	0.66
	✓	✓	0.69	**0.68**	**0.68**
✓	✓	✓	**0.70**	0.67	0.67

Table 2: Experimental results with the pre-trained BERT. P and R stand for precision and recall, respectively. All the reported numbers are macro-averaged metrics.

Among the baselines, the NumberBatch baseline, even with only three similarity numbers, achieved almost comparable performance with the BERT baseline. The NumberBatch baseline combined with the ARES baseline exhibited the best overall score with F1=0.68, whereas the full combination method displayed the best precision, 0.70. These results suggest that relatively concise features, in terms of dimensionality, are sometimes more effective than large-sized vector features. It implies that the full combination method could achieve better results with the properly dimensionality-reduced BERT vectors.

5.3. Results with Fine-tuned BERT

Table 3 presents the results, in which contextualized word representations of w_{l1} and w_{l2} obtained by the fine-tuned BERT (described in section 4.5) are used as features. As expected, the BERT and ARES baselines exhibited better results than those with the pre-trained BERT. In this setting, the full combination method achieved the best overall score with F1=0.67, which is, however, slightly worse than the best result of 0.68 achieved with the pre-trained BERT. These results should be more closely investigated, but it may suggest that the contextualized word representations fine-tuned with this particular task may be slightly incompatible with the NumberBatch-based features that refer to the WordNet graph structure and the static NumberBatch embeddings.

BERT	ARES	NumBat	P	R	F1
✓			0.67	0.66	0.66
	✓		0.65	0.65	0.64
✓	✓		0.67	0.66	0.66
	✓	✓	**0.68**	0.66	0.66
✓	✓	✓	0.67	**0.67**	**0.67**

Table 3: Experimental results with the fine-tuned BERT. The NumberBatch baseline is excluded from this table as it does not use BERT-originated vectors.

	Pre-trained BERT	Fine-tuned BERT
1	$sim_{CN}(u, v)$	Pearson's
2	KL-distance	Spearman's
3	cosine similarity	Euclid distance
4	Euclid distanc	KL-distance
5	$sim_{CN}(u, w_{1l})$	cosine similarity

Table 4: Importances of features. All but $sim_{CN}(u, v)$ and $sim_{CN}(u, w_{l1})$ are features used with the ARES baseline.

5.4. Feature Importances

By using the functionality provided by the Random Forest classifier, we can probe into the importance of each feature. Table 4 sorts the five most essential features in the ARES+NumberBatch combination method, where the second and third columns present the results

[10]The `AutoModelForSequenceClassification` class provided by the transformers library was used.

[11]We used the scikit-learn (`https://scikit-learn.org/`) toolkit.

with the pre-trained and fine-tuned BERT, respectively. As this table contrasts, the pre-trained BERT jointly uses both feature groups, while the fine-tuned BERT heavily relies on the ARES features.

5.5. Results with the Simple BERT-based Classifier

As described in section 4.5, we fine-tuned the BERT with a classification head with the current task; thereby, the trained classifier was directly used on its own in this task. Table 5 displays its performance. The first line shows the sentence pair-wise results, whereas the second line displays the results aggregated in the knowledge triple-chain, which could be compared with other baseline methods. In the aggregation process, a straight-forward majority voting on the sentence pair-wise results was conducted: If the voting for a knowledge triple chain ends in a tie, we predicted it as *with-gap*.

Level	P	R	F1
Sentence pair-wise	0.73	0.74	0.73
Chain-level	**0.70**	**0.70**	**0.70**

Table 5: Experimental results with the simple BERT-based classifier.

As shown in the table, this method achieved the best results of F1=0.70 amongst the other compared methods. The gap between the sentence pair-wise result and that of the chain-level suggests that we need to further stabilize the majority voting results. This could be accomplished by feeding more sentences with more variations into the BERT encoder. These results also suggest that the overall performance could be further enhanced by combining the BERT-based classifier with other classifiers that employ other features.

6. Related Work

Knowledge graph, in general, has been widely considered a vital ingredient to realizing intelligent systems in many fields of AI. Most systems using a knowledge graph rely on the paths retrieved from the graph to accomplish a kind of reasoning. Hence, selecting valid and better paths, given a particular target task, is a vital issue. Several attempts to validate a knowledge triple (Huaman et al., 2021) or to measure its trustworthiness (Jia et al., 2019) have been made. However, these works focus on fact-based knowledge graphs, not on a commonsense knowledge graph.

If we shift our focus to the topic of commonsense knowledge graphs, Zhou et al. (2019) proposed a learning-based method to predict ConceptNet path quality. They created a dataset maintaining crowdsourced human assessments of the "naturalness" of the sampled paths. One of the notable aspects of this work is that they did not give a concrete definition of naturalness to the crowd-workers, suggesting the difficulty of formally defining a valid commonsense knowledge path. It also highlights the differences between fact-based knowledge graphs with commonsense knowledge graphs. Davison et al. (2019) proposed a scheme for assessing the quality of a knowledge path, especially for evaluating the validity of a knowledge triple generated from a large pre-trained language model. Similar to our present work, they use relation-specific hand-crafted templates to generate sentences that are subsequently masked to train the model.

As discussed in this paper, our task is related to the ever-lasting issue of word sense disambiguation. As early as 2011, Chen and Liu (2011) tried to disambiguate ConceptNet nodes by measuring the relatednesses among the word groups that presumably associated with a sense of the target word. Our NumberBatch baseline is similar to their method in that both rely on a set of presumably related synsets or words.

We used the ARES sense embeddings (Scarlini et al., 2020), which is one of the representative sense embedding resources (Bevilacqua et al., 2021). Thanks to the nature of ARES embeddings that they can be directly compared with the BERT-based contextualized word embeddings, they are exceptionally convenient in a semantic task like the present work.

7. Concluding Remarks

This paper discussed a critical but not significantly discussed issue of semantic gaps that potentially exist in arbitrarily chosen paths in ConceptNet. To investigate the feasibility of a machine learning-based method, we created a pilot dataset in which the validity of each of the collected knowledge triple chains was human-annotated. In the preliminary experiments, we compared several baseline methods, which gave us several insights, including the potential efficacy of sense embeddings and contextualized word representations enabled by a pre-trained language model. However, the detection accuracies obtained with the presented methods are far from the desired level, implying that the target task itself is not trivial. As discussed in (Becker et al., 2019) that assessed the difficulty of ConceptNet relation classification, the difficulty of the present task may partly be attributed to the ontologically-loose natures of ConceptNet.

For future work, we would extend the initial dataset, preferably with crowdsourcing, and device better detection methods on top of it. We would achieve better-contextualized word representations by feeding more relation-specific sentences into the sentence encoder, which requires the improvement of the sentence generation templates. We will also implement a joint learning framework in which contextualized work representations work in tandem with the sense embeddings and other features. In parallel, we will work on a knowledge path evaluation method that effectively employs a version of the semantic gap detection method.

8. Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 22K12723.

9. Bibliographical References

Becker, M., Staniek, M., Nastase, V., and Frank, A. (2019). Assessing the difficulty of classifying ConceptNet relations in a multi-label classification setting. In *RELATIONS - Workshop on meaning relations between phrases and sentences*, Gothenburg, Sweden, May. Association for Computational Linguistics.

Bevilacqua, M., Pasini, T., Raganato, A., and Navigli, R. (2021). Recent trends in word sense disambiguation: A survey. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4330–4338.

Chen, J. and Liu, J. (2011). Combining ConceptNet and WordNet for word sense disambiguation. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 686–694, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Davis, E. and Marcus, G. (2015). Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103.

Davison, J., Feldman, J., and Rush, A. (2019). Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178, Hong Kong, China, November. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Christiane Fellbaum, editor. (1998). *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA.

Feng, Y., Chen, X., Lin, B. Y., Wang, P., Yan, J., and Ren, X. (2020). Scalable multi-hop relational reasoning for knowledge-aware question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1295–1309, Online, November. Association for Computational Linguistics.

Huaman, E., Tauqeer, A., and Fensel, A. (2021). Towards knowledge graphs validation through weighted knowledge sources. In Boris Villazón-Terrazas, et al., editors, *Knowledge Graphs and Semantic Web*, pages 47–60, Cham. Springer International Publishing.

Ilievski, F., Oltramari, A., Ma, K., Zhang, B., McGuinness, D. L., and Szekely, P. A. (2021). Dimensions of commonsense knowledge. *CoRR*, abs/2101.04640.

Jia, S., Xiang, Y., Chen, X., and Wang, K. (2019). Triple trustworthiness measurement for knowledge graph. *The World Wide Web Conference - WWW '19*.

Lin, B. Y., Chen, X., Chen, J., and Ren, X. (2019). KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839, Hong Kong, China, November. Association for Computational Linguistics.

Paul, D. and Frank, A. (2019). Ranking and selecting multi-hop knowledge paths to better predict human needs. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, volume 1, Minneapolis, Minnesota, USA.

Peters, W. and Peters, I. (2000). Lexicalised systematic polysemy in WordNet. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens, Greece, May. European Language Resources Association (ELRA).

Rashkin, H., Bosselut, A., Sap, M., Knight, K., and Choi, Y. (2018). Modeling naive psychology of characters in simple commonsense stories. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2289–2299, Melbourne, Australia, July. Association for Computational Linguistics.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In C. Raymond Perrault, editor, *Proceedings of the 14th IJCAI*, pages 448–453, Montréal (Canada).

Scarlini, B., Pasini, T., and Navigli, R. (2020). With more contexts comes better performance: Contextualized sense embeddings for all-round word sense disambiguation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3528–3539, Online, November. Association for Computational Linguistics.

Speer, R., Chin, J., and Havasi, C. (2017). Conceptnet 5.5: An open multilingual graph of general knowledge. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb.

Talmor, A., Herzig, J., Lourie, N., and Berant, J. (2019). CommonsenseQA: A question answering challenge targeting commonsense knowledge. In

Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4149–4158, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Zhou, Y., Schockaert, S., and Shah, J. (2019). Predicting conceptnet path quality using crowdsourced assessments of naturalness. *The World Wide Web Conference on - WWW '19*.

Appendix

A: Relation-specific Checking Rules

Table 6 summarizes the relation-specific checking rules for flagging presumably invalid knowledge triples. The column "p1=p2" presents a part-of-speech constraint: "y" requires the parts-of-speech of the head and tail should be identical. The columns "p1" and "p2" respectively constrain the part-of-speech that the head/tail of a knowledge triple should carry. We consult WordNet to retrieve the possible set of parts-of-speech for a word. If none of them match the constraints specified by the relevant rule, the corresponding knowledge triple is flagged invalid in advance to the human annotation process.

B: Detailed Experimental Results

Tables 7 and 8 respectively display the detailed experimental results obtained with three classifiers. Note that each of the tables corresponds to the summarized Tables 2 or 3. Each block in these tables consists of three rows, respectively presenting the result with Logistic Regression (L), Random Forest (R), and Support Vector Machine (S). Recall that the summarized Tables 2 and 3 only display the results with the RF classifier.

Relation	p1 = p2	p1	p2
AtLocation	-	-	n
CapableOf	-	n	v
Causes	-	-	-
CausesDesire	-	-	-
CreatedBy	-	n	-
DefinedAs	-	-	-
Desires	-	n	-
Entails	y	-	-
HasA	y	n	n
HasFirstSubevent	y	v	v
HasLastSubevent	y	v	v
HasPrerequisite	-	-	-
HasProperty	-	n	a
HasSubevent	y	v	v
InstanceOf	y	n	n
IsA	y	-	-
LocatedNear	-	-	n
MadeOf	y	n	n
MannerOf	-	-	v
MotivatedByGoal	-	v	-
NotCapableOf	-	n	v
NotDesires	-	n	-
NotHasProperty	-	n	a
PartOf	y	n	n
ReceivesAction	-	n	v
SimilarTo	-	-	-
SymbolOf	-	-	-
Synonym	y	-	-
UsedFor	-	-	-

Table 6: Relation-specific checking rules. n, v, and a respectively denote parts-of-speech, noun, verb, and adjective.

	BERT	ARES	NumBat	P	R	F1
L	✓			0.62	0.62	0.62
R	✓			0.68	0.65	0.65
S	✓			0.67	0.66	0.65
L		✓		0.64	0.63	0.62
R		✓		0.64	0.61	0.60
S		✓		0.64	0.62	0.62
L			✓	0.66	0.66	0.66
R			✓	0.66	0.65	0.65
S			✓	0.66	0.66	0.66
L	✓	✓		0.67	0.66	0.66
R	✓	✓		0.69	0.65	0.66
S	✓	✓		0.67	0.66	0.66
L		✓	✓	0.68	0.68	0.68
R		✓	✓	0.69	0.68	0.68
S		✓	✓	0.68	0.68	0.68
L	✓	✓	✓	0.66	0.65	0.65
R	✓	✓	✓	0.70	0.67	0.67
S	✓	✓	✓	0.68	0.67	0.67

Table 7: Detailed experimental results with the pre-trained BERT. The first column indicates the classifier employed: L, R, and S respectively denote Logistic Regression, Random Forest, and Support Vector Machine. All the reported numbers are macro-averaged metrics.

	BERT	ARES	NumBat	P	R	F1
L	✓			0.67	0.67	0.67
R	✓			0.67	0.66	0.66
S	✓			0.67	0.67	0.67
L		✓		0.62	0.62	0.62
R		✓		0.65	0.65	0.64
S		✓		0.62	0.63	0.62
L	✓	✓		0.67	0.67	0.67
R	✓	✓		0.67	0.66	0.66
S	✓	✓		0.67	0.67	0.67
L		✓	✓	0.67	0.66	0.66
R		✓	✓	0.68	0.66	0.66
S		✓	✓	0.67	0.66	0.66
L	✓	✓	✓	0.68	0.67	0.68
R	✓	✓	✓	0.67	0.67	0.67
S	✓	✓	✓	0.67	0.67	0.67

Table 8: Detailed experimental results with the fine-tuned BERT. The NumberBatch baseline is excluded from this table, as it does not use BERT embeddings.