2023

IERL: Interpretable Ensemble Representation Learning - Combining CrowdSourced Knowledge and Distributed Semantic Representations

Yuxin Zi
University of South Carolina - Columbia

Kaushik Roy

Vignesh Narayanan
University of South Carolina - Columbia

Manas Gaur

Amit Sheth
University of South Carolina - Columbia

Publication Info

Preprint version 2023.

IERL: Interpretable Ensemble Representation Learning - Combining CrowdSourced Knowledge and Distributed Semantic Representations

Yuxin Zi[1], Kaushik Roy[1], Vignesh Narayanan[1], Manas Gaur[2], and Amit Sheth[1]

[1] Artificial Intelligence Institute, University of South Carolina
{vignar,amit}@sc.edu
{yzi,kaushikr}@email.sc.edu
[2] University of Maryland, Baltimore County
manas@umbc.edu

Abstract. Large Language Models (LLMs) encode meanings of words in the form of distributed semantics. Distributed semantics capture common statistical patterns among language tokens (words, phrases, and sentences) from large amounts of data. LLMs perform exceedingly well across General Language Understanding Evaluation (GLUE) tasks designed to test a model's understanding of the meanings of the input tokens. However, recent studies have shown that LLMs tend to generate unintended, inconsistent, or wrong texts as outputs when processing inputs that were seen rarely during training, or inputs that are associated with diverse contexts (e.g., well-known *hallucination* phenomenon in language generation tasks). Crowdsourced and expert-curated knowledge graphs such as ConceptNet are designed to capture the meaning of words from a compact set of well-defined contexts. Thus LLMs may benefit from leveraging such knowledge contexts to reduce inconsistencies in outputs. We propose a novel ensemble learning method, the *Interpretable Ensemble Representation Learning* (IERL), that systematically combines LLM and crowdsourced knowledge representations of input tokens. IERL has the distinct advantage of being interpretable by design (when was the *LLM context* used vs. when was the *knowledge context* used?) over state-of-the-art (SOTA) methods, allowing scrutiny of the inputs in conjunction with the parameters of the model, facilitating the analysis of models' inconsistent or irrelevant outputs. Although IERL is agnostic to the choice of LLM and crowdsourced knowledge, we demonstrate our approach using BERT and ConceptNet. We report improved or competitive results with IERL across GLUE tasks over current SOTA methods and significantly enhanced model interpretability.

Keywords: Knowledge Graphs · Language Models · Knowledge Infusion · Model Interpretability · GLUE Tasks

1 Introduction

LLMs have performed exceedingly well on the GLUE benchmark tasks [1]. GLUE tasks measure the machine's comprehension on supervised learning-based natural language processing tasks, such as Quora Question Pairs to check question redundancy, and Recognizing Textual Entailment to check if two sentences share entailment, neutral, or contraction relations [2]. LLMs learn trillions of parameters after training over a humongous amount of data. Irregularities in the data (for example, little or highly varying language token patterns or contexts) causes LLMs to *hallucinate* - generating inconsistent outputs for similar inputs.

Crowdsourced and curated knowledge graphs (KGs), such as ConceptNet, are designed to capture meanings of commonly used words using a compact set of contexts agreed by humans [3,4]. As a result, representations learned from ConceptNet are less likely to suffer from distributional irregularities among the tokens. Consequently, it is a promising and an active research topic on utilizing representations from KGs to potentially mitigate irregularities, while processing input tokens via LLM representations. In this paper, we focus on developing a learning method that systematically incorporates representations from both KGs and LLMs to address the following unresolved questions. **Q1**: *Can we design an approach to combine crowdsourced knowledge and LLM representations to obtain an integrated representation, in order to mitigate the model hallucination?* **Q2**: *Can we achieve an interpretable design - i.e., can we tractably discern for what inputs the LLM hallucinates on and what knowledge context improves representation quality?* Next, we briefly review existing methods that seek to infuse representations from KGs and LLMs and their relevance to questions **Q1** and **Q2**.

1.1 Related Work on Combining Knowledge and LLM Representations

There is an extensive literature on combining LLMs and knowledge representations to leverage contextual information among language tokens from both [5]. The representations are then processed through a task-specific neural network. Here we will cover the four SOTA approaches, KALA, K-Adapter, TDLR, and GCT, broadly representing two kinds of methods - (1) Combining representations at the input level before passing it through the neural network (KALA and K-Adapter) and (2) Combing representations at the parametric level, i.e., modify the parameters of the task-specific neural network and the resulting token meaning interpretations (TDLR and GCT) [6,7,8,9].

KALA modifies the input LLM representations for tokens by using a weighted aggregate of other tokens connected in the knowledge graph. K-Adapter trains "adapter" models for encoding knowledge representations and combines the LLM and adapter representations at the input level. With KALA and K-Adapter, it is not possible to keep track of and understand how the representations are incorporated into the neural network after the input stage internally. Ablation

studies and post-hoc approximate interpretations using LIME, etc., provide representation interpretability (was the knowledge context important or the data context?)[10]. However, it is unclear how far the approximation is off from the truth, which is crucial to evaluating and systematically addressing LLM hallucination issues. Furthermore, the effect of KALA and K-Adapter representations on hallucinations has not been studied. Thus, KALA and K-Adapter do not fully address **Q1** or **Q2**. TDLR operates on the self-attention mechanism of transformers by modifying the attention or weight matrices to hard-code graph connections among language tokens. TDLR does this once, in the first self-attention block, and then allows model fine-tuning to continue as is. It is unclear if the attention matrix modification is retained during the fine-tuning across the remaining transformer blocks. GCT is similar to TDLR and differs in the specifics of the self-attention matrix modification operation. TDLR and GCT suffer from similar issues as KALA and K-Adapter towards addressing **Q1** and **Q2**.

2 Background and Motivation

In this section, we describe the GLUE tasks, what hallucination looks like when solving the GLUE tasks,, and the theoretical motivations for the IERL algorithm presented in Section 3.

2.1 Task Descriptions and Hallucinations

We experiment with similarity or entailment GLUE tasks that take a pair of sentences as input, and format its output as a $+1$ or a -1. For the similarity tasks, $+1$ and -1 correspond to similar or dissimilar input sentences respectively. For the entailment tasks, $+1$ and -1 correspond to entailment and contradiction, respectively.

We use X to denote the dataset, and x is an instance in the dataset X. Each x is a three-tuple composed of $x[1]$: sentence 1, $x[2]$: sentence 2, and the label y.

Hallucinations. Hallucinations refers to inconsistent model outputs for similar inputs resulting from statistical irregularities in the data. Since this notion is often used in the context of language generation tasks, we clarify the context in which we use it in this paper. To formalize this notion, we batch our instances into random batches. We note the convergence rate variations of the training loop across these batches, where each batch is of size equal to 80% of the training dataset X. We find that for the GLUE tasks, SOTA fine-tuning results in a high convergence rate variance (ranges from $13 - 45$ iterations). This suggests that there may be a high degree of irregularity in the statistical properties across the batches. Therefore, we can expect a model trained on these datasets to generate inconsistent outputs for similar inputs, i.e., suffer from hallucinations.

2.2 Ensemble Learning Approach

To fine-tune the models for GLUE tasks, a few feedforward neural network layers are added and trained using backpropagation. Such a training procedure works well when there is a generalizable pattern across the instances in the fine-tuning dataset (regularly occurring statistical patterns). To tackle the issue of irregularities, we propose using example patterns from each instance and aggregating them using an ensemble learning approach. We can think of an example pattern as one that maps a given instance to its output in the task-specific dataset, which can be seen as (an instance level) model and define an ensemble function $g(z)$ for a new point instance z as an ensemble of weighted contributions from similar instances in the dataset X as

$$g(z) = \sum_{x} (\alpha^1_{x[1]} \odot \langle z, x[1] \rangle + \alpha^2_{x[2]} \odot \langle z, x[2] \rangle). \tag{1}$$

Here $\odot$ refers to a product operation defined in the algorithm and experimentation sections, and $\langle \cdot \rangle$ refers to a suitable similarity computation.

2.3 Utilizing Knowledge Graph Contexts

The ensemble approach formulation allows the expressiveness to model both generalizability across instances and instance-level details. Examining the ensemble model's parameters lets us interpret whether an instance shows irregular patterns. However, it does not yet incorporate a mechanism to solve the irregularity issue during model learning. Here we posit that combining LLM and knowledge graph representations using an ensemble approach allows us to interpret instances for their pattern regularities and draw from either the LLM or knowledge contexts to solve the irregularity resulting in high performance that hallucinates less.

Thus, we expand the formulation in (1) to describe the operation of combining LLM and knowledge representations as

$$g(z) = \sum_{x} (\alpha^{sim}_{LLM} \cdot [\langle z, x[1] \rangle^{sim}_{LLM} \ \langle z, x[2] \rangle^{sim}_{LLM}] + \alpha^{sim}_{KG} \cdot [\langle z, x[1] \rangle^{sim}_{KG} \ \langle z, x[2] \rangle^{sim}_{KG}]) \tag{2}$$

Here, $\cdot$ refers to the dot product between vectors, $\langle \cdot \rangle^{sim}_{LLM}$ refers to a similarity measure between the LLM representations, and $\langle \cdot \rangle^{sim}_{KG}$ refers to a similarity measure between the KG embedding representations. The α^{sim}_{LLM} and α^{sim}_{KG} are two dimensional vectors.

Aggregation Methods Aggregation as a method to reduce statistical irregularities such as high variance has been well-studied in statistical learning theory literature [11]. The ensemble formulation in (2) can be seen as aggregation over instances in the dataset X. In this work, we experiment with two types of aggregation, the average of the instance representations and using averages over higher-order moment representations. We can expect LLMs trained on very

large amounts of data to tend to the *normal* distributional trend in the underlying data distribution. Therefore the average (first-order moment) and variance (second-order moment) of groups of instance representations are *sufficient statistics* to describe the underlying distribution. However, for smaller number of data instances (such as in GLUE task datasets), it may be necessary to utilize averages over higher order moments as *sufficient statistics*. We experiment with both types of aggregation and compare the results.

3 The Interpretable Ensemble Representation Learning (IERL) Algorithm

Figure 1 shows an illustration of the IERL optimization step - **(a)** Shows the dataset X (e.g., Recognizing Textual Entailment) and its instances x_i indexed by i. $t_i[1]$ and $t_i[2]$ denote the BERT representations of sentence 1: $x_i[1]$ and 2: $x_i[2]$ from instance i. $c_i[1]$ and $c_i[2]$ denote the ConceptNet representations of sentences 1 and 2 from instance i. **(b)** Shows how similar and dissimilar instances to $x_i[1]$ are constructed and aggregated for the cases of $y_i == 1$ and -1 respectively. **(c)** Shows one step of optimization in detail corresponding to line 22 in Algorithm 3 **(d)** Shows two methods of aggregation over instances - Averaging and Moment-Based (Algorithm 2) aggregation. Algorithm 1 and 2 detail the IERL and aggregation algorithm, respectively.

4 Experimental Section and Results

In our experiments, we use BERT as the choice of LLM representations and ConceptNet NumberBatch embeddings for the choice of KG representations in the IERL algorithm (3). We use gradient descent as our optimization procedure, and use grid search to tune hyperparameters for optimization. For the computation of higher order moment representations in algorithm 2, we execute each for loop iteration in parallel. We initialize the parameters $\alpha_i[j]$ for each (i, j) using a 0 mean, I covariance 4d-gaussian distribution. We test our method on the GLUE tasks: Quora Question Pairs (QQP), Question-Answering NLI (QNLI), Multi-Genre NLI (MNLI), RTE, Stanford Sentiment Treebank v2 (SST-2), and Winograd NLI (WNLI) pertaining to two types of tasks:

1. **Sentence Similarity**: Consists of input sentence pairs and a 1 or 0 denoting if the pairs are similar or not (we reformulate to 1 and -1) - QQP, and STS
2. **Sentence Entailment**: Consists of input sentence pairs and label from among "entailment, contradiction, neutral" (we reformulate to 1 for entailment and -1 for contradiction) - QNLI, WNLI, MNLI, and RTE

We also convert our vectors to unit vectors before computing dot products (line 21 in IERL Algorithm - 3).

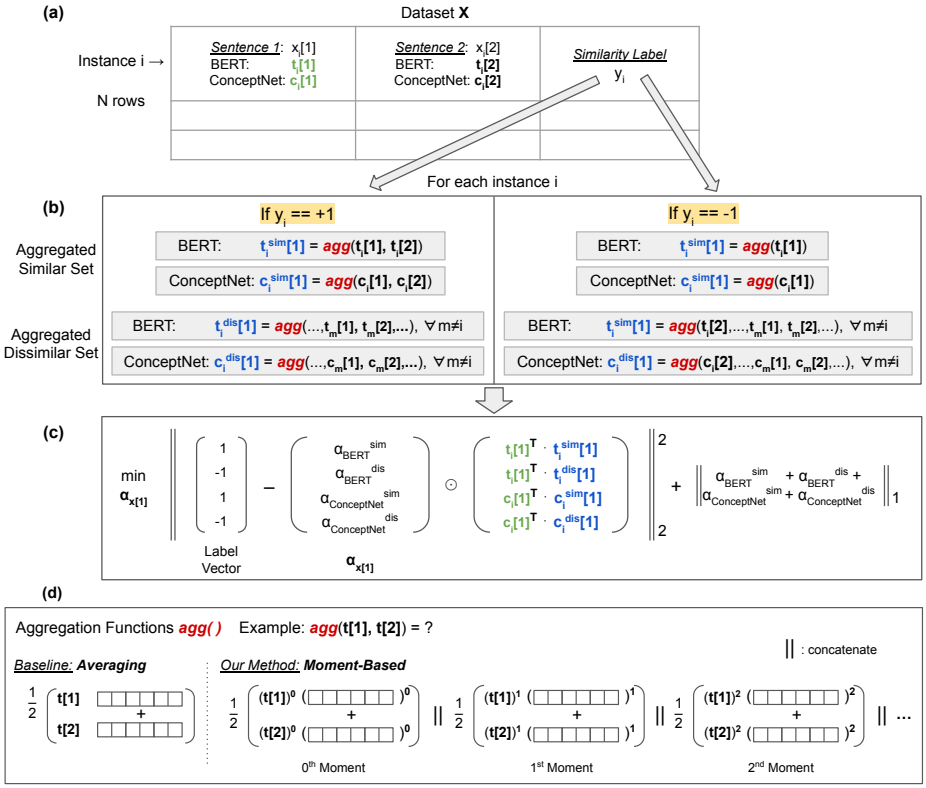

Fig. 1. (a) Shows the dataset X (e.g., Recognizing Textual Entailment (RTE)) and its instances x_i indexed by i ($i = 1, \ldots, N$). The BERT representations of sentence 1: $x_i[1]$ and 2: $x_i[2]$ from instance i are denoted by $t_i[1]$ and $t_i[2]$. The ConceptNet representations of sentences 1 and 2 from instance i are denoted by $c_i[1]$ and $c_i[2]$, respectively. **(b)** Shows how similar and dissimilar instances to $x_i[1]$ are constructed and aggregated for the cases of $y_i = 1$ and -1, respectively. **(c)** Shows one of the optimization problems in detail corresponding to line 22 in algorithm 3. **(d)** Shows two methods of aggregation over instances - Averaging and Moment-Based (algorithm 2) aggregation.

Baseline Model For our baseline model we implement IERL using a simple average for aggregation (instead of computing moments using Algorithm 2). We call this $\text{IERL}_{\text{BASE}}$. We present our evaluation results in the order that they address the questions **Q1** and **Q2** introduced in section 1.

4.1 Quantitative Evaluation - Addresses Q1

We report accuracy measures of our method against the baseline $\text{IERL}_{\text{BASE}}$ and the current leader on the GLUE leaderboard and see that the performance of IERL shows competitive performance even against state-of-the-art performance.

Algorithm 1 Interpretable Ensemble Representation Learning (IERL)

Inputs: Dataset: $x \in X$ ▷ $x = (sentence1, sentence2, label)$, see section 2.1
Models = { } ▷ Initialize dictionary to store models for all sentences in X
for $i \in X$ **do** ▷ i indexes instance x
  $x_i = X[i]$ ▷ ith instance
  $t_i[1],\ t_i[2] = LLM(x_i[1]),\ LLM(x_i[2])$ ▷ LLM representations for sentences
  $c_i[1],\ c_i[2] = KG(x_i[1]),\ KG(x_i[2])$ ▷ KG representations for sentences
  $y = x[3]$ ▷ Label for this instance
  for $j \in \{1, 2\}$ **do** ▷ j indexes each sentence in the instance
    if y == +1 **then** ▷ Similar and dissimilar instance aggregation
      $t_i^{sim}[j]$ =agg($[t_i[1],\ t_i[2]]$)
      $c_i^{sim}[j]$ =agg($[c_i[1],\ c_i[2]]$)
      $t_i^{dis}[j]$ =agg($[.., t_m[1], t_m[2], ..]), \forall m \neq i$
      $c_i^{dis}[j]$ =agg($[.., c_m[1], c_m[2], ..]), \forall m \neq i$
    else if y == -1 **then** ▷ Similar and dissimilar instance aggregation
      $t_i^{sim}[j]$ =agg($[t_i[1]]$)
      $c_i^{sim}[j]$ =agg($[c_i[1]]$)
      $t_i^{dis}[j]$ =agg($[t_i[2], .., t_m[1], t_m[2], ..]), \forall m \neq i$
      $c_i^{dis}[j]$ =agg($[c_i[2], .., c_m[1], c_m[2], ..]), \forall m \neq i$
    $\alpha_i[j] = [\alpha_{LLM}^{sim}, \alpha_{LLM}^{dis}, \alpha_{KG}^{sim}, \alpha_{KG}^{dis}]$ ▷ parameters for instance i, sentence j
    $I = [1, -1, 1, -1]$
    $D = [t_i[j] \cdot t_i^{sim}[j], t_i[j] \cdot t_i^{dis}[j], c_i[j] \cdot c_i^{sim}[j], c_i[j] \cdot c_i^{dis}[j]]$
    Optimization until convergence: Minimize $g_i[j] = ||I - \alpha_i[j] \odot D||_2^2 + ||\alpha_i[j]||_1^1$
    Models$[x_i[j]] = (g_i[j], \alpha_i[j])$ ▷ Store model for instance i, sentence j
return Models

Algorithm 2 Aggregation Algorithm (agg)

Inputs: list of vectors V
$agg_V = [\ ]$
for $v \in V$ **do** ▷ calculate element wise powers of the vector elements
  $v_0, v_1, v_2, v_3 = v^0, v^1, v^2, v^3$
  $v_{concat} = concat(v_0, v_1, v_2, v_3)$ ▷ Concatenate the power vectors
  $agg_V.append(v_{concat})$ ▷ add to list of vectors to aggregate
return $mean(agg_V)$ ▷ return average of all lists in agg_V

We also compute #Optimization steps using randomly sampled batches of size 80% of the whole dataset per sample and tabulate the range (min-max)[3]. We see that the range is significantly higher using an implementation of BERT (BERT (Ours) - Vanilla BERT with 6 layers and fine-tuning) compared to both versions of IERL. Furthermore, higher-order moments also show a much faster convergence of 7-13 steps vs. 20-30 and 20-45 steps. We use up to fourth-order moments in our experiments, i.e., 0-3.

[3] We are currently running fine-tuning using the leaderboard model and will report #Optimization-Steps range in future work

System	STS	QQP	QNLI	WNLI	MNLI	RTE	#Optimization-Steps (Range)
$\text{GLUE}_{\text{LEADER}}$	93.5	90.9	96.7	97.9	92.5	93.6	-
BERT (Ours)	89.7	88.7	93.5	93.3	81.5	88.3	20-45
$\text{IERL}_{\text{BASE}}$	90.89	86.41	92.3	90.11	88.53	90.4	20-30
IERL	**93.55**	**90.51**	**95.56**	**98.7**	**92.08**	**92.3**	**7-13**

Table 1. Comparing IERL performance on similarity and entailment GLUE tasks. We also see that the # of Optimization steps stabilizes using the IERL training method. IERL shows competitive performance even against state-of-the-art performance. Using higher-order moments in IERL shows a much faster convergence of 7-13 steps vs. 20-45 and 20-30 steps.

4.2 Qualitative Evaluation - Addresses Q2

Figure 2 shows an example inference output using IERL for a group of test sentences and an anchor sentence z (z chosen for ease of illustration). The figure shows a group of instances shown in the oval and rectangular boxes (including z) and similarity measurements. For a pair of instances z and one other instance from the group shown (let it be denoted by $z2$), we first find the closest sentences x_1, x_2 from the training set X and compute two similarities as $s1 = \hat{BERT}(x_1) \cdot \hat{BERT}(x_2)$ and $s2 = \hat{ConceptNet}(x_1) \cdot \hat{ConceptNet}(x_2)$, where $\hat{(.)}$ represents normalizing the vectors as unit vectors. We display the greater of the two. The shapes are highlighted in green when the sum of the similarities is greater than or equal to $Models[x_1] \cdot Models[x_2]$, i.e., inference value = 1 (line 23 in algorithm 3) and highlighted in pink otherwise, i.e., inference value = -1. The rectangular shape denotes the $s1 \geq s2$, and the oval shape denotes that $s2 \geq s1$ (the parameter values also reflect the same in α_{x1} and α_{x2}). Thus IERL is designed to provide a simple method to interpret the inference results for a group of test sentences.

5 Conclusion and Future Work

In this work, we propose Interpretable Ensemble Representation Learning (IERL) as an ensemble technique that demonstrates the interpretable combination of LLM and knowledge representations to result in a high-performance model that is robust to hallucinations and results in faster convergence in the number of optimization steps. Through our experiments, we see the promise of IERL as a method that advances research towards combining LLMs and knowledge graphs that retain both high performances and are interpretable by design (thus, addressing interpretability ambiguities during ablations and approximate post-hoc interpretations). In future work, we will explore different LLM and KG choices and vary the order of moments considered. Furthermore, we will explore other naturally interpretable combination functions (e.g., linear combination ensemble in this work) that can add layers of expressiveness to the interpretation (e.g., abstraction level in a hierarchy of concepts from a KG).

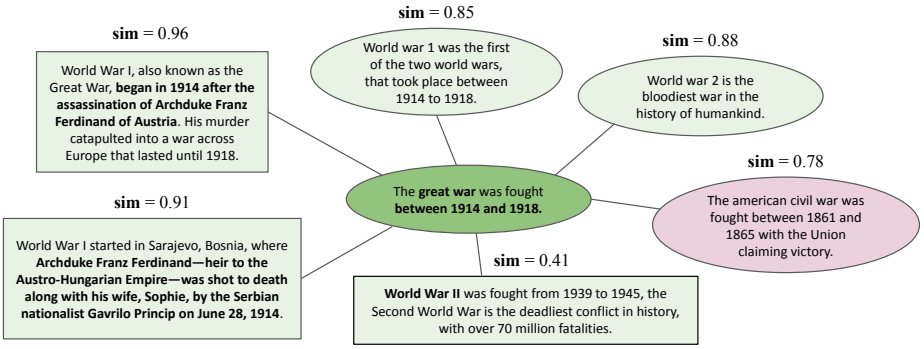

Fig. 2. Shows an example inference output using IERL for a group of test sentences along with an anchor sentence z (z chosen for ease of illustration). The figure shows a group of instances shown in the oval and rectangular boxes (including z) and similarity measurements. For a pair of instances z and one other instance from the group shown (let it be denoted by $z2$), we first find the closest sentences x_1, x_2 from the training set X and compute two similarities as $s1 = \hat{BERT}(x_1) \cdot \hat{BERT}(x_2)$ and $s2 = \hat{ConceptNet}(x_1) \cdot \hat{ConceptNet}(x_2)$, where $\hat{(.)}$ represents normalizing the vectors as unit vectors. We display the greater of the two. The shapes are highlighted in green when the sum of the similarities is greater than or equal to $Models[x_1] \cdot Models[x_2]$, i.e., inference value = 1 (line 23 in algorithm 3) and highlighted in pink otherwise, i.e., inference value = -1. The rectangular shape denotes the $s1 \geq s2$, and the oval shape denotes that $s2 \geq s1$ (the parameter values also reflect the same in α_{x1} and α_{x2}). Thus IERL is designed to provide a simple method to interpret the inference results for a group of test sentences.

References

1. Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.
2. Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.
3. Phillip Bricker. Ontological commitment. 2014.
4. Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*, 2017.
5. Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2021.
6. Minki Kang, Jinheon Baek, and Sung Ju Hwang. Kala: Knowledge-augmented language model adaptation. *arXiv preprint arXiv:2204.10555*, 2022.
7. Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Guihong Cao, Daxin Jiang, Ming Zhou, et al. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*, 2020.

8. Vipula Rawte, Megha Chakraborty, Kaushik Roy, Manas Gaur, Keyur Faldu, Prashant Kikani, Hemang Akbari, and Amit P Sheth. Tdlr: Top semantic-down syntactic language representation. In *NeurIPS'22 Workshop on All Things Attention: Bridging Different Perspectives on Attention*.
9. Edward Choi, Zhen Xu, Yujia Li, Michael Dusenberry, Gerardo Flores, Emily Xue, and Andrew Dai. Learning the graphical structure of electronic health records with graph convolutional transformer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 606–613, 2020.
10. Saumitra Mishra, Bob L Sturm, and Simon Dixon. Local interpretable model-agnostic explanations for music content analysis. In *ISMIR*, volume 53, pages 537–543, 2017.
11. William AV Clark and Karen L Avery. The effects of data aggregation in statistical analysis. *Geographical Analysis*, 8(4):428–438, 1976.