Zero-Shot Learning with Common Sense Knowledge Graphs

Nihal V. Nayak
Department of Computer Science
Brown University
nihal_nayak@brown.edu

Stephen H. Bach
Department of Computer Science
Brown University
stephen_bach@brown.edu

Abstract

Zero-shot learning relies on semantic class representations such as attributes or pretrained embeddings to predict classes without any labeled examples. We propose to learn class representations from common sense knowledge graphs. Common sense knowledge graphs are an untapped source of explicit high-level knowledge that requires little human effort to apply to a range of tasks. To capture the knowledge in the graph, we introduce ZSL-KG, a framework based on graph neural networks with non-linear aggregators to generate class representations. Whereas most prior work on graph neural networks use linear functions to aggregate information from neighboring nodes, we find that non-linear aggregators such as LSTMs or transformers lead to significant improvements on zero-shot tasks. On two natural language tasks across three datasets, ZSL-KG shows an average improvement of 9.2 points of accuracy versus state-of-the-art methods. In addition, on an object classification task, ZSL-KG shows a 2.2 accuracy point improvement versus the best methods that do not require hand-engineered class representations. Finally, we find that ZSL-KG outperforms the best performing graph neural networks with linear aggregators by an average of 3.8 accuracy points across these four datasets.

1 Introduction

In recent years, deep neural networks have achieved impressive performance on a range of tasks in language, vision and speech [20, 52, 73]. These techniques require a huge amount of labeled training data to achieve optimal performance. This is a severe bottleneck as obtaining large amounts of hand-labeled data is an expensive process. Zero-shot learning is a training strategy which allows a machine learning model to predict novel classes without the need for any labeled examples for the new classes [41, 48, 55]. Zero-shot learning models learn parameters for seen classes along with their class representations. During inference, new class representations are provided for the unseen classes. Previous zero-shot learning systems have used attributes [2, 10, 23], pretrained embeddings [11] and learnable embeddings (e.g. sentence embeddings) [60] as class representations.

We need to consider several factors while designing a zero-shot learning framework: (1) it should adapt to unseen classes without requiring additional human effort, (2) it should provide rich features such that the unseen classes have sufficient distinguishing characteristics among themselves, (3) it should be applicable to a range of downstream tasks.

Previous approaches for class representations have various limitations. On one end of the spectrum, attribute-based methods provide rich features but curating attributes for each class is a cumbersome process and the attributes have to be decided ahead of time for the unseen classes. On the other end of the spectrum, pretrained embeddings such as GloVe [37] and Word2Vec [32] offer the flexibility of easily adapting to new classes but rely on unsupervised training on large corpora—which may not provide distinguishing characteristics necessary for zero-shot learning. Recent approaches using

Preprint. Under review.

graph neural networks for zero-shot object classification have achieved state-of-the-art performance [17, 56]. GCNZ [56] and DGP [17] train a graph neural network on the ImageNet graph to generate class representations. However, in Section 4 we show that these methods often do not perform well beyond object classification.

In our work, we propose to learn class representations from common sense knowledge graphs. Common sense knowledge graphs [27, 49, 50, 71] represent high-level knowledge implicit to humans as concept nodes. These graphs have explicit edges between related concept nodes and provide valuable information to distinguish between different concepts. However, adapting existing zero-shot learning frameworks to learn class representations from common sense knowledge graphs can be problematic in several ways. GCNZ [56] learns graph neural networks with a symmetrically normalized graph Laplacian, which not only requires the entire graph structure during training but also needs retraining if the graph structure changes. DGP [17] aims to generate expressive class representations and uses an asymmetrically normalized graph Laplacian but assumes a directed acyclic graph such as WordNet.

To address these limitations, we propose ZSL-KG, a new framework based on graph neural networks. Graph neural networks learn to represent the structure of graphs by aggregating information from each node's neighbourhood. Aggregation techniques used in GCNZ, DGP, and most other graph neural network approaches are linear, in the sense that they take a (possibly weighted) mean or maximum of the neighbourhood features. On the other hand, non-linear aggregators such as LSTMs [16] can potentially generate more expressive features. Some recent works [13, 33] that have considered LSTM aggregators achieved competitive performance in comparison to linear aggregators for relatively low-dimensional tasks such as node classification. We find that non-linear graph aggregators based on LSTMs and transformers are particularly beneficial for zero-shot learning. However, an LSTM is not uniformly the best performer. To address this challenge and make ZSL-KG more flexible, we introduce a transformer [52] aggregator for graph neural networks. Additionally, our framework is inductive, i.e., the graph neural network can be executed on graphs that are different from the training graph, which is necessary for inductive zero-shot learning under which the test classes are unknown during training.

We demonstrate the effectiveness of our framework on three zero-shot learning tasks in language and vision: intent classification, fine-grained entity typing, and object classification. On the language tasks, we observe an average improvement of 9.2 points of accuracy on three datasets over the existing state-of-the-art benchmarks. On object classification, we find a 2.2 accuracy point improvement versus the best methods that do not require hand-engineered class representations [17, 56]. Finally, we perform a study on the choice of neighbourhood aggregator in our ZSL-KG framework. We find that non-linear aggregators perform significantly better than linear aggregators on zero-shot learning by an average 3.8 accuracy points. In summary, our main contributions are the following:

1. We propose learning zero-shot class representations from common sense knowledge graphs.
2. We present ZSL-KG, a framework for for zero-shot learning based on graph neural networks with non-linear aggregators. To increase the flexibility of ZSL-KG, we introduce a novel transformer-based aggregator for graph neural networks.
3. ZSL-KG achieves new state-of-the-art scores on language tasks for the SNIPS-NLU [7], FIGER [25], and Ontonotes [12] datasets. It also achieves a new state-of-the art score among methods that do not use hand-engineered class representations on object classification for Animals with Attributes 2 [61], and is 1 point away from the overall state-of-the-art.

2 Background

In this section, we summarize zero-shot learning and graph neural networks.

2.1 Zero-Shot Learning

Zero-shot learning has several variations [55, 61]. Our work focuses on inductive zero-shot learning, under which we do not have access to the unseen classes during training. Given a training set $D = (x_i, y_i)$ with $y_i \in Y_S$ the set of seen classes, we aim to learn a classifier $f : X \to Y$. Unlike traditional classification tasks, zero-shot learning systems are trained along with class representations such as attributes, pretrained embeddings, etc.

Recent approaches learn a class encoder $\phi(y) \in \mathbb{R}^d$ to produce vector-valued class representations from an initial input, such as a string or other identifier of the class. (In our case, y is a node in a graph and its k-hop neighborhood.) During inference, the class representations are used to label examples with the unseen classes by passing the examples through an example encoder $\theta(x) \in \mathbb{R}^d$ and predicting the class whose representation has the highest inner product with the example representation.

Recent work in zero-shot learning commonly uses one of two approaches to learn the class encoder $\phi(y)$. One approach uses a bilinear similarity function defined by a compatibility matrix $W \in \mathbb{R}^{d\times d}$ [11, 61]:

$$f\left(\theta(x), \mathbf{W}, \phi(y)\right) = \theta(x)^{\mathrm{T}}\mathbf{W}\phi(y)\ . \tag{1}$$

The bilinear similarity function gives a score for each example-class pair. The parameters of θ, $\mathbf{W}$, and ϕ are learned by taking a softmax over f for all possible seen classes $y \in Y_S$ and minimizing either the cross entropy loss or a ranking loss with respect to the true labels. In other words, f should give a higher score for the correct class(es) and lower score for the incorrect classes. $\mathbf{W}$ is often constrained to be low rank, to reduce the number of learnable parameters [35, 69]. Lastly, other variants of the similarity function add minor variations such as non-linearities between factors of $\mathbf{W}$ [48, 60].

The other common approach is to first train a neural network classifier in a supervised fashion. The final fully connected layer of this network has a vector representation for each seen class, and the remaining layers are used as the example encoder $\theta(x)$. Then, the class encoder $\phi(y)$ is trained by minimizing the L2 loss between the representations from supervised learning and $\phi(y)$ [17, 48, 56].

The class encoder that we propose in Section 3 can be plugged into either approach.

2.2 Graph Neural Networks

The basic idea behind graph neural networks is to learn node embeddings that reflects the structure of the graph [14]. Consider the graph $G = (V, E, R)$, where V is the set of vertices with node features X_v and $(v_i, r, v_j) \in E$ are the labeled edges and $r \in R$ are the relation types. Graph neural networks learn node embeddings by iterative aggregation of the k-hop neighbourhood. Each layer of a graph neural network has two main components $\mathrm{AGGREGATE}$ and $\mathrm{COMBINE}$ [63]:

$$\mathbf{a}_v^{(l)} = \mathrm{AGGREGATE}^{(l)}\left(\left\{\mathbf{h}_u^{(l-1)} \forall u \in \mathcal{N}(v)\right\}\right) \tag{2}$$

where $\mathbf{a}_v^{(l)} \in \mathbb{R}^{d_{l-1}}$ is the aggregated node feature of the neighbourhood, $\mathbf{h}_u^{(l-1)}$ is the node feature in neighbourhood $\mathcal{N}(.)$ of node v. The aggregated node is passed to the $\mathrm{COMBINE}$ to generate the node representation $\mathbf{h}_v^{(l)} \in \mathbb{R}^{d_l}$ for the l-th layer:

$$\mathbf{h}_v^{(l)} = \mathrm{COMBINE}^{(l)}\left(\mathbf{h}_v^{(l-1)}, \mathbf{a}_v^{(l)}\right) \tag{3}$$

$\mathbf{h}_v^{(0)} = \mathbf{x}_v$ where $\mathbf{x}_v$ is the initial feature vector for the node. Previous works on graph neural networks for zero-shot learning have used GloVe [37] to represent the initial features [17, 56].

3 Zero-shot Learning with Common Sense Knowledge Graphs

Here we introduce ZSL-KG: a framework based on graph neural networks with common sense knowledge graphs for zero-shot learning. We first highlight the issues while modeling common sense knowledge graphs. Then, we address these challenges with inductive graph neural networks with non-linear aggregators.

Common sense knowledge graphs organize high-level knowledge implicit to humans in a graph. The nodes in the graph can be abstract concepts as well as definitional concepts. For example, the concept `politician` is associated with definitional concepts such as `mayor` and `statesman` as well as common sense concepts `running_town`, `town_hall`, etc. The nodes are linked with edges associated with relation types. For instance, `(mayor, IsA, politician)` has the relation `IsA` linking `mayor` to `politician`. These associations in the graph offer a rich source of information, which makes common sense knowledge graphs applicable to a wide range of tasks. However, because

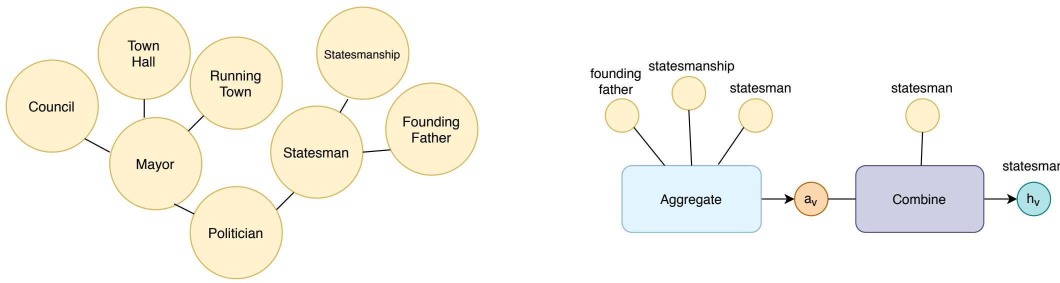

Figure 1: *Left:*A sample from the 2-hop neighbourhood for the concept `politician` from ConceptNet. *Right:*The figure describes the aggregate and the combine function used in our graph neural network. The neighbourhood nodes are passed through the non-linear aggregate function to generate the aggregated node feature. The aggregated node feature is passed to the combine function to generate the node embedding. Optionally, we can pass the node's feature into the combine function from the previous layer which serves as a form of residual connection. These operations are recursively computed for k-hops to generate the class representation.

they are rich, they are also large scale. Publicly available common sense knowledge graphs range roughly from 100,000 to 8 million nodes and 2 million to 21 million edges [49, 71].

To learn class representations from common sense knowledge graphs, we look to graph neural networks. Existing methods from zero-shot object classification such as GCNZ and DGP cannot be adapted to common sense knowledge graphs as they do not scale to large graphs or require a directed acyclic graph. Other general-purpose graph neural networks that use linear aggregators to learn the structure of the graph might be inadequate to capture the complex relationships in the knowledge graph. Recent work on LSTM aggregators has shown strong performance on low-dimensional task such as node classification, but has not been explored for complex zero-shot learning tasks [33].

We propose to learn class representations with non-linear graph aggregators: either LSTMS or a novel approach using transformers. Figure 1 shows the high-level architecture of our aggregator.

LSTM Aggregator. LSTMs [16] are used for learning sequences—often in language tasks—where they take sequence of words or features as input to make predictions. This means that LSTMs are not permutation invariant i.e. the order the node features can affect the output. To overcome this limitation, we randomly permute the neighbourhood nodes while generating the aggregated vector $\mathbf{a}_v^{(l)}$. The nodes in the neighbourhood are passed through the LSTM to obtain the aggregated vector $\mathbf{a}_v^{(l)}$. Then, concatenate the aggregated vector $\mathbf{a}_v^{(l)}$ with the node's feature from the previous layer $\mathbf{h}_v^{(l-1)}$ and pass through the linear layer followed by a non-linearity $\sigma(.)$.

Formally:

$$\mathbf{a}_v^{(l)} = \mathrm{LSTM}^{(l)}\left(\mathbf{h}_u^{(l-1)} \forall u \in \mathcal{N}(v) \cup \{v\}\right) \quad \mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \cdot \mathrm{CONCAT}[\mathbf{h}_v^{(l-1)}, \mathbf{a}_v^{(l)}]\right) \tag{4}$$

where $\mathrm{LSTM}^{(l)}$ is the learnable LSTM network and $\mathbf{W}^{(l)} \in \mathbb{R}^{2d_{(l-1)} \times d_{(l)}}$ is a learnable projection weight matrix for the l-th layer of the graph neural network.

Transformer Aggregator. Transformers [52] take the entire input sequence to generate hidden states for each input feature. We take advantage of this property to achieve non-linear aggregation of the neighbourhood features. To make transformers permutation invariant, we simply do not add the sinusoidal positional embedding to the input. The nodes in the neighbourhood are passed through the transformer, the non-linear aggregator, to obtain the aggregated vector $\mathbf{a}_v^{(l)}$. The aggregated vector $\mathbf{a}_v^{(l)}$ is passed through a linear layer followed by a non-linearity $\sigma(.)$

Formally:

$$\mathbf{a}_v^{(l)} = \mu\left[\mathrm{TRANSFORMER}^{(l)}\left(\left\{\mathbf{h}_u^{(l-1)} \forall u \in \mathcal{N}(v) \cup \{v\}\right\}\right)\right] \quad \mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \cdot \mathbf{a}_v^{(l)}\right) \tag{5}$$

where $\text{TRANSFORMER}^{(l)}$ is learnable transformer, $\mu(.)$ is a pooling function such as mean-pooling which combines the node features to get aggregated vector $\mathbf{a}_v^{(l)}$ and $\mathbf{W} \in \mathbb{R}^{d_{(l-1)} \times d_{(l)}}$ is a learnable projection weight matrix.

Variants of graph neural networks that use relational information to learn the graph structure such as RGCN use linear aggregators [31, 44]. In section 4.4, we show that the LSTM and transformer aggregators significantly outperform other existing relational graph neural networks.

In our experiments, we use ConceptNet [49] as our common sense knowledge graph but our approach can be adapted to other knowledge graphs. ConceptNet is a large scale knowledge graph with millions of nodes and edges, which poses a challenge to train the graph neural network. To solve this problem, we explored numerous neighbourhood sampling strategies. Existing work on sampling neighbourhood includes random sampling [13], importance sampling [4], random walks [68], etc. Similar to PinSage [68], we simulate random walks for the nodes in the graph and assign hit probability to the neighbourhood nodes. During training and testing the graph neural network, we select the top N nodes from the neighbourhood based on their hitting probability.

4 Tasks and Results

We evaluate our framework on three zero-shot learning tasks: intent classification, fine-grained entity typing, and object classification. First, we describe the general setup and preprocessing of ConceptNet. Next, we detail how we adapt zero-shot object classification methods to our tasks. Then, we introduce the individual tasks, datasets and report results on each of the task. Finally, we perform an ablation on the choice of neighbourhood aggregators in our framework. The code and hyperparameters are included in the supplementary material, which will be released upon acceptance.

ConceptNet Setup. In all our experiments, we map each class to a node in ConceptNet 5.7 [49] and query its 2-hop neighbourhood. For example, a class `politician` gets mapped to `/c/en/politician` in the graph and its 2-hop neighbourhood is queried. Then, we remove all the non-English concepts and their edges from the graph and make all the relations bidirectional. For fine-grained entity typing and object classification, we also take the union of the concepts' neighbourhood that share the same prefix. For example, we take the union of the `/c/en/politician` and `/c/en/politician/n`. Then, we compute the embeddings for the concept using the pretrained GloVe 840B [37]. We average the individual words in the concept to get the embedding. These embeddings serve as initial features for the graph neural network. We ran the random walks on train and test classes separately so no information about the identity of the test classes leaked.

Experimental Setup. We evaluate ZSL-KG with an LSTM aggregator (ZSL-KG-LSTM) and a Transformer aggregator (ZSL-KG-Tr). We use two-layer graph neural networks, corresponding to two hops around each class node. During training and testing, we pick the 50 and 100 node neighbours with the highest hitting probabilities for the first and the second hop, respectively.

We adapt GCNZ, SGCN, and DGP for zero-shot learning tasks in language. GCNZ [56] uses symmetrically normalized graph Laplacian to generate the class representations. SGCN [17] uses an asymmetrical normalized graph Laplacian to learn the class representations. Finally, DGP [17] exploits the hierarchical graph structure and avoids dilution of knowledge from intermediate nodes. They use a dense graph connectivity scheme with a two-stage propagation from ancestors and descendants to learn the class representations. We mapped the classes to the nodes in the WordNet graph for each dataset. On the language tasks, all three methods use two-layer graph neural networks. On the computer vision task, we use the code obtained from [17] to replicate the methods.

4.1 Intent Classification

Intent Classification is the task of identifying users' intent expressed in chatbots and personal voice assistants. Developing zero-shot learning systems for intent classification can allow existing models to adapt to emerging intents in personal assistants.

Dataset. We evaluate on the main open-source benchmark for intent classification: SNIPS-NLU [7]. The dataset was collected using crowdsourcing to benchmark the performance of voice assistants.

Experiment. Intent classification is a zero-shot multi-class classification task. Since the number of classes is small, we use the bilinear similarity architecture for zero-shot learning. The example

Methods	Accuracy
DeViSE [11]	74.47
IntentCapsNet [59]	77.52
ReCapsNet-ZS [28]	79.96
GCNZ [56]	82.47 ± 03.09
SGCN [17]	50.27 ± 14.13
DGP [17]	64.41 ± 12.87
ZSL-KG-LSTM	88.81 ± 01.17
ZSL-KG-Tr	**88.98** ± **01.22**

Table 1: Results for intent classification on the SNIPS-NLU dataset. We report the average performance of the models on 5 random seeds and the standard error. The results for DeViSE, IntentCapsNet and ReCapsNet-ZS are obtained from [28]

encoder is a BiLSTM with attention. The words in the example are represented with GloVe 840B. The training set has 5 classes which we split into 3 train classes and 2 development classes. We trains for 10 epochs by minimizing the cross entropy loss and pick the model with the least loss on the development set. We measure prediction accuracy.

We compare ZSL-KG against existing state-of-the-art approaches in the literature for intent classification: DeViSE [11], IntentCapsNet [59], and ResCapsNet-ZS [28]. DeViSE uses pretrained embeddings as class representations. IntentCapsNet and ResCapsNet-ZS are CapsuleNet [43] based approaches and have reported the best performance on the task.

Results. Table 1 shows the results. Both variants of our framework significantly outperform the existing approaches and improves the state-of-the-art accuracy to 88.98%. The results for the baselines from zero-shot object classification indicates that GCNZ perform slightly better than existing benchmarks. Finally, SGCN and DGP performs poorly and shows high variance on the task.

4.2 Fine-Grained Entity Typing

Fine-grained entity typing is the task of classifying named entities into one or more narrowly scoped semantic types. Identifying fine-grained types of named entities has shown to improve to downstream performance in relation extraction [64], question answering [66] and coreference resolution [9].

Datasets. We evaluate on two popular fine-grained entity typing datasets: FIGER [25] and OntoNotes [12]. The datasets are collected using a combination of distant supervision from knowledge bases and heuristics to label the entities.

Both datasets are traditionally used in a supervised setting. OTyper [70] created a zero-shot learning dataset for FIGER, where they divided their classes into 10 folds of train, development and test classes. Similar to FIGER, we convert Ontonotes into a zero-shot learning dataset and split the classes into multiple folds. See Section B.2 for more details.

Experiment. We reconstructed OTyper [70] and DZET [35], the state-of-the-art methods for this task. Both methods use the AttentiveNER biLSTM [47] as the example encoder. See Section B.1 for more details. Otyper averages the GloVe embeddings for the words in the name of each class to represent it. For DZET, we manually mapped the classes to Wikipedia articles. We pass each article's first paragraph through a learnable biLSTM to obtain the class representations.

We train each model for 5 epochs by minimizing the cross-entropy loss and pick the model with the least loss on the development set. We evaluate the performance on the unseen classes by computing the micro average strict accuracy and macro average strict accuracy across all the folds. See Section B.3 for more details.

Results. Table 2 shows results of experiments on fine-grained entity typing. Our results show that ZSL-KG-LSTM significantly outperforms the state-of-the-art methods. ZSL-KG-Tr does not perform as well as ZSL-KG-LSTM on the task, but is also competitive with the state of the art. We also observe that methods from zero-shot object classification do not perform as well on fine-grained entity typing compared to other existing specialized methods from the language community.

	FIGER		*Ontonotes*	
	Mic. Avg.	Mac. Avg.	Mic. Avg.	Mac. Avg.
Methods				
OTyper [70]	54.39 ± 1.97	58.08 ± 0.75	33.38 ± 0.02	31.67 ± 1.44
DZET [35]	50.12 ± 1.48	50.15 ± 1.49	37.37 ± 3.95	34.49 ± 3.61
GCNZ [56]	37.43 ± 1.78	48.41 ± 1.13	38.07 ± 1.71	40.22 ± 1.11
SGCN [17]	41.58 ± 1.02	53.92 ± 1.78	22.15 ± 1.27	29.21 ± 1.91
DGP [17]	39.58 ± 1.26	49.55 ± 1.60	27.38 ± 0.52	31.29 ± 1.31
ZSL-KG-LSTM	**68.95 ± 1.49**	**65.57 ± 1.30**	**44.66 ± 2.40**	**42.72 ± 1.95**
ZSL-KG-Tr	52.37 ± 2.75	54.96 ± 1.46	34.46 ± 2.25	32.63 ± 2.07

Table 2: The results for zero-shot fine-grained entity typing on FIGER and Ontonotes. We report the average performance of the models on 5 random seeds and the standard error.

Methods	Accuracy
GCNZ [56]	70.7
SGCN [17]	74.24 ± 1.67
DGP [17]	74.30 ± 1.10
ZSL-KG-LSTM	65.22 ± 1.03
ZSL-KG-Tr	**76.50 ± 0.67**

Table 3: Results for object classification on the AWA2 dataset. We report the average performance of the models on 5 random seeds and the standard error. The results for GCNZ are obtained from [17].

4.3 Object Classification

To assess ZSL-KG's versatility, we also consider object classification, a computer vision task.

Datasets. We evaluate on the Animals with Attributes 2 (AWA2) dataset [61]. AWA2 contains images of animals with 40 classes in the train and validation sets, and 10 classes in the test set. Each class is annotated with 85 attributes, such as whether it as stripes, lives in water, etc.

Experiment. Following prior work [17, 56], here we use the L2 loss architecture for zero-shot learning. The example encoder and seen class representations come from the ResNet 50 model [15] in Torchvision [30] pretrained on ILSVRC 2012 [42]. We map the the ILSVRC 2012 training and validation classes, and the AWA2 test classes to ConceptNet. The model is trained on 950 random classes and the remaining 50 ILSVRC 2012 the classes are used for validation. We use the same setting for SGCN and DGP using the authors' implementation. The model with the least loss on the validation classes is used to make predictions on the test classes. Again following prior work, we predict on the 10 test classes of the updated split [61] and report averaged per-class accuracy.

Results. Table 3 shows the results for zero-shot object classification. ZSL-KG-Tr outperforms all other existing methods which learn class representations with graph neural networks. While ZSL-KG-LSTM performs relatively poorly on the task, it was clear from the validation loss during traing that ZSL-KG-Tr should be preferred on this task. ZSL-KG sets a new state of the art among methods which do not use the hand-engineered class attributes, and is 1 point away from the highest reported score among all methods [54].

4.4 Comparison of Graph Aggregators

We conduct an ablation study with different aggregators with our framework. Existing the graph neural networks include - GCN [19], GAT [53], and RGCN [44]. GCN [19] computes the mean of the node neighbours to learn the graph structure. GAT [53] computes the edge attention and applies it to the node features to learn the structure of the graph. RGCN [44] conditions the neighbourhood features with a learnable weight and then applies a weighted mean to learn the graph structure. We provide all the architectural details in section C. We train these models with the same experimental setting for the tasks mentioned in their respective sections.

Results. Table 4 shows results for our ablation study. Our results show that ZSL-KG outperforms existing graph neural networks with linear aggregators. With relational aggregators, we observe that

	SNIPS-NLU	*FIGER*	*Ontonotes*	*AWA2*
	Accuracy	Mic. Avg.	Mic. Avg.	Accuracy
GCN [19]	84.78 ± 0.77	52.71 ± 2.64	33.11 ± 1.68	73.68± 0.63
GAT [53]	87.57 ± 1.59	56.71 ± 1.07	31.12 ± 1.97	74.70± 0.78
RGCN [44]	87.47 ± 1.81	65.60 ± 1.90	35.67 ± 5.55	65.10± 1.01
ZSL-KG-LSTM	88.81 ± 1.17	**68.95 ± 1.49**	**44.66 ± 2.40**	65.22± 1.03
ZSL-KG-Tr	**88.98 ± 1.22**	52.37 ± 2.75	34.46 ± 2.25	**76.50 ± 0.67**

Table 4: The results for zero-shot learning tasks with other graph neural networks. We report the average performance on 5 random seeds.

they do not outperform non-linear aggregators and may reduce the overall performance (as seen in AWA2). Finally, our results on intent classification suggests that common sense knowledge graphs with even simple linear aggregators can outperform existing state-of-the-art benchmark results [28].

5 Related Works

We broadly describe the related works for zero-shot learning and graph neural networks.

Zero-Shot Learning. Zero-shot learning has been thoroughly researched in the computer vision community for object classification [2, 10, 11, 23, 55, 61]. Recent works in zero-shot learning have used graph neural networks for object classification [17, 56]. In our work, we extend their approach to general-purpose common sense knowledge graphs to generate class representations. Other notable works, use generative methods for generalized zero-shot learning where both seen and unseen classes are evaluated at test time [21, 45]. But, these methods still rely on hand-crafted attributes for classification. Zero-shot learning for text classification is a well-studied problem [8, 34, 36, 72]. Previously, ConceptNet has been used for transductive zero-shot text classification as a shallow soruce of knowledge for class represention [72]. They use ConceptNet to generate a sparse vector which is combined with pretrained embeddings and description to obtain the class representation. On the other hand, we use ConceptNet to generate a dense vector representations from a graph neural network and use them as our class representation. Other line of work treats zero-shot text classification as a textual entailment problem and benchmarks their performance on several large-scale text classification dataset [67]. Fine-grained entity typing [6, 12, 25, 47, 69] has been thoroughly researched for several years. However, limited approaches are tackling zero-shot fine-grained entity typing due to the lack of formalism in the task. Existing methods use different datasets and vary the evaluation metric. Most recent works in zero-shot learning for fine-grained entity typing have used a bilinear similarity model with a different class representation [29, 35, 70].

Graph Neural Networks. Recent works on graph neural networks have demonstrated significant improvements for several downstream tasks such as node classification and graph classification [13, 14, 19, 53, 57]. Extensions of graph neural networks to relational graphs have produced significant results in several graph related tasks[31, 44, 46, 51]. Training on large graphs is a challenging task and existing works in the field have explored sampling techniques to achieve the same performance with a subsampled graph [4, 5, 74, 68]. We use a random walk based approach to train our graph neural network. Furthermore, several diverse applications using graph neural networks have been explored: common sense reasoning [24], fine-grained entity typing [62], text classification [65], reinforcement learning [1] and neural machine translation [3]. For a more in-depth review, we point readers to [58].

6 Conclusion

In conclusion, we present ZSL-KG, a general-purpose framework based on graph neural networks to learn class representations from common sense knowledge graphs. We show that our framework outperforms existing state-of-the-art benchmarks in two challenging zero-shot learning tasks in language. Furthermore, we demonstrate the versatility of framework by beating the best methods that do not require hand-engineered class representation. Finally, our study comparing our framework with other linear graph neural networks shows that non-linear graph aggregators perform significantly better linear aggregators.

Broader Impact

Deep neural networks require huge amounts of labeled training data to achieve strong performance. Labeling data is an expensive process as it requires humans to manually annotate the datasets. Our work on zero-shot learning aims to reduce the cost of labeling datasets. Our framework can be integrated into existing weak supervision pipelines such as Snorkel[39], where ZSL-KG can be used as a source of supervision along with other labeling functions. Our framework can also be used to provide initial labels for active learning pipelines and can be continually improved with feedback.

Although reducing the cost of labeling data can appear to be a positive effect, many human annotators rely on labeling data as a source of income. Automating the labeling pipelines has a negative impact on their livelihood. But, jobs in the past have been automated by technology. For instance, ATMs automated the job of a cashier to a great extent. We suspect that the role of human annotators could a temporary phase during this machine learning boom and will eventually cease to exist. Furthermore, historical evidence suggests that technological automation in the past has not lead to long-term effects on the unemployment rates [38].

Our system can fail in two ways: (1) the zero-shot class is not a node in the graph (2) the zero-shot prediction is incorrect. In the first case, our model simply cannot predict and new nodes/edges need to be added to the graph for the model to work. However, the second case presents a greater ethical concern where our model's predictions are incorrect. Our current system does not offer any explanation when the model predicts an incorrect label for an example. A possible future work could be to receive explanations for the predictions and use humans to make informed decisions in critical applications [22, 26, 40].

There is a growing interest in the community to represent 'implicit' common sense knowledge as graphs. Our work hinges on the correctness of the knowledge graph. Common sense knowledge graphs are usually constructed by aggregating existing knowledge graphs and through crowdsourcing. This process can include offensive terms and associations in the knowledge graph, which can affect the downstream applications. For example, in figure 1, we can see that `politician` has an edge to `statesman`. This could introduce gender-related bias into our fine-grained entity typing model. Curating the knowledge graph by filtering offensive and biased nodes can be a possible solution. For instance, Conceptnet 5.8 filtered edges from their graph using metadata from wiktionary to reduce offensive terms. They also report no significant drop in performance on semantic benchmarks as these edges were not valuable.

Acknowledgements

We thank Yang Zhang for help preparing the Ontonotes dataset. We thank Roma Patel and Elaheh Raisi for providing helpful feedback on our work. This material is based on research sponsored by Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL) under agreement number FA8750-19-2-1006. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL) or the U.S. Government. We gratefully acknowledge support from Google. Disclosure: Stephen Bach is an advisor to Snorkel AI, a company that provides software and services for weakly supervised machine learning.

References

[1] A. Adhikari, X. Yuan, M.-A. Côté, M. Zelinka, M.-A. Rondeau, R. Laroche, P. Poupart, J. Tang, A. Trischler, and W. L. Hamilton. Learning dynamic knowledge graphs to generalize on text-based games. *arXiv preprint arXiv:2002.09127*, 2020.

[2] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2015.

[3] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an. Graph convolutional encoders for syntax-aware neural machine translation. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[4] J. Chen, T. Ma, and C. Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *International Conference on Learning Representations (ICLR)*, 2018.

[5] J. Chen, J. Zhu, and L. Song. Stochastic training of graph convolutional networks with variance reduction. *International Conference on Machine Learning (ICML)*, 2018.

[6] E. Choi, O. Levy, Y. Choi, and L. Zettlemoyer. Ultra-fine entity typing. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018.

[7] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *ICML workshop on Privacy in Machine Learning and Artificial Intelligence*, 2018.

[8] Y. N. Dauphin, G. Tur, D. Hakkani-Tur, and L. Heck. Zero-shot learning for semantic utterance classification. *arXiv preprint arXiv:1401.0509*, 2013.

[9] G. Durrett and D. Klein. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the association for computational linguistics (TACL)*, 2014.

[10] A. Farhadi, I. Endres, D. Hoiem, and D. A. Forsyth. Describing objects by their attributes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[11] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[12] D. Gillick, N. Lazic, K. Ganchev, J. Kirchner, and D. Huynh. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*, 2014.

[13] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[14] W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 2017.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[17] M. Kampffmeyer, Y. Chen, X. Liang, H. Wang, Y. Zhang, and E. P. Xing. Rethinking knowledge graph propagation for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[19] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[21] V. Kumar Verma, G. Arora, A. Mishra, and P. Rai. Generalized zero-shot learning via synthesized examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[22] V. Lai and C. Tan. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *Proceedings of FAT**, 2019.

[23] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2014.

[24] B. Y. Lin, X. Chen, J. Chen, and X. Ren. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[25] X. Ling and D. S. Weld. Fine-grained entity recognition. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2012.

[26] Z. C. Lipton. The mythos of model interpretability. *ACM Queue*, 2018.

[27] H. Liu and P. Singh. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 2004.

[28] H. Liu, X. Zhang, L. Fan, X. Fu, Q. Li, X.-M. Wu, and A. Y. Lam. Reconstructing capsule networks for zero-shot intent classification. 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)*, 2019.

[29] Y. Ma, E. Cambria, and S. Gao. Label embedding for zero-shot fine-grained named entity typing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016.

[30] S. Marcel and Y. Rodriguez. Torchvision: The machine-vision package of torch. In *International Conference on Multimedia*, 2010.

[31] D. Marcheggiani and I. Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[32] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[33] R. L. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *International Conference on Learning Representations (ICLR)*, 2019.

[34] J. Nam, E. L. Mencía, and J. Fürnkranz. All-in text: Learning document, label, and word representations jointly. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

[35] R. Obeidat, X. Fern, H. Shahbazi, and P. Tadepalli. Description-based zero-shot fine-grained entity typing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[36] N. Pappas and J. Henderson. Gile: A generalized input-label embedding for text classification. *Transactions of the Association for Computational Linguistics*, 2019.

[37] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014.

[38] C. Pissarides. Over half of the world's jobs are replaceable by a robot - is yours?, 2019.

[39] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *The VLDB Journal*, 2020.

[40] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016.

[41] B. Romera-Paredes and P. Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning (ICML)*, 2015.

[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.

[43] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[44] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, 2018.

[45] E. Schonfeld, S. Ebrahimi, S. Sinha, T. Darrell, and Z. Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[46] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[47] S. Shimaoka, P. Stenetorp, K. Inui, and S. Riedel. Neural architectures for fine-grained entity type classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers (EACL)*, 2017.

[48] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[49] R. Speer, J. Chin, and C. Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[50] N. Tandon, G. De Melo, and G. Weikum. Webchild 2.0: Fine-grained commonsense knowledge distillation. In *Proceedings of ACL 2017, System Demonstrations*, 2017.

[51] S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[53] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *International Conference on Learning Representations (ICLR)*, 2018.

[54] V. K. Verma, D. Brahma, and P. Rai. A meta-learning framework for generalized zero-shot learning. *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[55] W. Wang, V. W. Zheng, H. Yu, and C. Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2019.

[56] X. Wang, Y. Ye, and A. Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[57] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger. Simplifying graph convolutional networks. *International Conference on Machine Learning (ICML)*, 2019.

[58] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[59] C. Xia, C. Zhang, X. Yan, Y. Chang, and P. S. Yu. Zero-shot user intent detection via capsule neural networks. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[60] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele. Latent embeddings for zero-shot classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[61] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2018.

[62] W. Xiong, J. Wu, D. Lei, M. Yu, S. Chang, X. Guo, and W. Y. Wang. Imposing label-relational inductive bias for extremely fine-grained entity typing. *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) (NAACL)*, 2019.

[63] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[64] Y. Yaghoobzadeh and H. Schütze. Corpus-level fine-grained entity typing using contextual information. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[65] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[66] S. Yavuz, I. Gur, Y. Su, M. Srivatsa, and X. Yan. Improving semantic parsing via answer type inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[67] W. Yin, J. Hay, and D. Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. 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)*, 2019.

[68] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[69] D. Yogatama, D. Gillick, and N. Lazic. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 2015.

[70] Z. Yuan and D. Downey. Otyper: A neural architecture for open named entity typing. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[71] H. Zhang, D. Khashabi, Y. Song, and D. Roth. Transomcs: From linguistic graphs to commonsense knowledge. *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[72] J. Zhang, P. Lertvittayakumjorn, and Y. Guo. Integrating semantic knowledge to tackle zero-shot text classification. 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)*, 2019.

[73] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[74] D. Zou, Z. Hu, Y. Wang, S. Jiang, Y. Sun, and Q. Gu. Few-shot representation learning for out-of-vocabulary words. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Example	*Class label*
Can you please look up the game the islanders ?	Search
Is it going to get chillier at 5 am in trussville bosnia and herzegovina ?	Weather
Add song to my club hits	Playlist

Table 5: Samples from the SNIPS-NLU dataset

Example	*Dataset*	*Class labels*
Valley Federal Savings & Loan Association said Imperial Corp. of America withdrew from regulators its application to buy five Valley Federal branches, leaving the transaction in limbo.	OntoNotes	/organization /organization/company
The students have been working for the IPNW clinic since this summer and have had multiple work sessions with **Rep. Tina Orwall**, who sponsors the compensation statute.	FIGER	/person /person/politician

Table 6: Samples from the OntoNotes and FIGER dataset for fine-grained entity typing. The **bold text** represents the entity and the class labels point to its fine-grained types.

A Dataset Details

Here, we provide additional details and examples from the datasets in our experiments.

A.1 Intent Classification

In our intent classification experiments, we benchmark our results on the SNIP-NLU dataset. Table 5 shows samples from the SNIPS-NLU dataset.

A.2 Fine-grained entity typing

Table 6 shows examples from the fine-grained entity typing datasets. FIGER dataset has 113 classes in the dataset. We pick 41 classes and divide them into 10 folds of train, development and test sets [70]. During training, we remove classes from the examples which are not included in the training split for the fold. Similarly, Ontonotes (after postprocessing in section B.2) has 86 classes in the dataset. Then, we pick 35 classes and divide them into 7 folds of train, development and test sets.

B Fine-grained Entity Typing Details

B.1 AttentiveNER

We describe AttentiveNER that we use to represent the example in fine-grained entity typing task. Each mention m comprises of n tokens which are individually mapped to their respective pretrained word embedding. We average these embeddings to obtain a single vector v_m. Formally, we can define v_m vector as:

$$v_m = \frac{1}{n}\sum_{j=1}^{n} m_j \tag{6}$$

where $m_j \in \mathbb{R}^d$ and $v_m \in \mathbb{R}^d$. We also need to learn the context of the mention. The left context l is represented by $\{l_1, l_2, ..., l_s\}$ and the right context r by $\{r_1, r_2, ..., r_s\}$ where $l_i \in R^d$ and $r_j \in \mathbb{R}^d$ are the word embeddings for the left and the right context respectively and s is the window size for the context. We pass l and r to the BiLSTM separately to obtain the hidden vectors $\overleftarrow{h_l}$, $\overrightarrow{h_l}$ for the left context and $\overleftarrow{h_r}$, $\overrightarrow{h_r}$ for the right context.

We then pass the hidden vectors to the attention layer. The attention layer is a 2 layer feedforward neural network and computes the attention for each of the hidden vectors. The attention values are normalized and used to compute the weighted sum of the hidden vectors to obtain the context vector $v_c \in \mathbb{R}^h$. Formally, we describe the operations below:

$$e_i^l = tanh(W_e \begin{bmatrix} \overleftarrow{h_i^j} \\ \overrightarrow{h_i^j} \end{bmatrix});$$

$$\alpha_i^l = W_\alpha e_i^l$$

We normalize the scalar attention values:

$$a_i^l = \frac{\alpha_i^l}{\sum_i \alpha_i^l + \sum_j \alpha_j^r}$$

The scalar values are multiplied with their respective hidden vectors to get the final context vector representation v_c:

$$v_c = \sum_{i=0}^{s} a_i^l h_i^l + \sum_{j=0}^{s} a_j^r h_j^r$$

Apart from the mention representation and context representation, we learn the hand-crafted feature representation v_f from the mention's syntactic, word-form and topic features. Finally, we concatenate the context vector v_c, v_f and v_m to get the input representation x.

B.2 OntoNotes Setup

We observed several challenges with Ontonotes. Ontonotes has the class /other associated with the named-entities, making the named-entity ambiguous. For example, "Ducks were the most precious asset in our village." and "A kitchen knife ; a knife from your kitchen at home." – the entity Ducks and knife are labeled with /other. Furthermore, it is not clear which is the correct concept for /other label as it an all encompassing word. To solve this issue, we remove /other labels from the dataset and rename all its subclasses without the /other prefix. We choose 35 classes from the list of 86 labels as the test classes. Like FIGER, the 35 classes are split into 7 folds of test for zero-shot learning.

B.3 Evaluation of fine-grained entity typing

Our fine-grained entity typing setup has multiple folds for OntoNotes and FIGER. Furthermore, the task is a multi-label classification problem. Stemming from existing research in fine-grained entity typing [25], we modify the strict accuracy metric. We introduce micro average strict accuracy and macro average strict accuracy to evaluate the performance of our model across multiple folds.

Micro average strict accuracy =

$$\frac{\sum_{f=1}^{F} \sum_{m \in M_f} \mathrm{I}(y_m = \hat{y}_m)}{\sum_{f=1}^{F} |M_f|} \tag{7}$$

and Macro average strict accuracy =

$$\frac{1}{|F|} \sum_{f=1}^{F} \frac{\sum_{m \in M_f} \mathrm{I}(y_m = \hat{y}_m)}{|M_f|} \tag{8}$$

where F corresponds to the folds, M_f are examples in the test set with unseen classes for the fold and $\mathrm{I}(.)$ is the indicator function.

Method	*Aggregate*	*Combine*
GCN	$\mathbf{a}_v^{(l)} = \text{Mean}\left(\left\{\mathbf{h}_u^{(l-1)}, u \in \mathcal{N}(v) \cup \{v\}\right\}\right)$	$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)}\mathbf{a}_v^{(l)}\right)$
GAT	$\alpha_u^{(l)} = \text{Attn}\left(\left\{(\mathbf{h'}_u^{(l-1)} \vert\vert \mathbf{h'}^{(l-1)})_v, u \in \mathcal{N}(v) \cup \{v\}\right\}\right)$	$\mathbf{h}_v^{(l)} = \sigma(\sum_1^{\mathcal{N}(v)+1} \alpha_u^{(l)} \mathbf{h'}_u^{(l-1)})$
RGCN	$\mathbf{a}_v^{(l)} = \sum_{r \in R} \sum_{j \in N(v)^r} \frac{1}{c_{i,r}} \sum_{b \in B} \alpha_{b,r}^{(l)} V_b^{(l)} \mathbf{h}_j^{(l-1)}$	$\mathbf{h}_v = \sigma(\mathbf{a}_v + \mathbf{W}_s^{(l)} \mathbf{h}_v^{(l-1)})$

Table 7: Graph Aggregators

C Graph Neural Networks Architecture Details

In our work, we compare ZSL-KG framework with multiple graph neural network architectures, namely - GCN, GAT and RGCN.

GCN uses a mean aggregator to learn the neighbourhood structure. GAT projects the neighbourhood nodes to a new features $\mathbf{h'}_u^{(l-1)} = \mathbf{W}\mathbf{h}_u^{(l-1)}$. The neighbourhood node features are concatenated with self feature and passed through a self-attention module for get the attention coefficients. The attention coefficients are multiplied with the neighbourhood features to the get the node embedding for the l-th layer in the combine function. RGCN uses a relational aggregator to learn the structure of the neighbourhood. To avoid overparameterization from the relational weights, we perform basis decomposition of the weight vector into B bases. We learn $|B|$ relational coefficients and $|B|$ weight vectors in the aggregate function and add with the self feature in combine function.

D Hyperparameters

In this section, we detail the hyperparameters used in our experiments.

D.1 Training Details

Our framework is built using PyTorch and AllenNLP[1]. In all our experiments, we use Adam [18] to train our parameters with a learning rate of 0.001. For intent classification, we experiment with a weight decay of 1e-05 and 5e-05. We found that weight decay of 5e-05 gives the best performance overall in intent classification for all the baseline graph aggregators. In intent classification, ZSL-KG-LSTM and ZSL-KG-Tr use weight decay of 5e-05 and 1e-05 respectively. We add a weight decay of 1e-05 for the OntoNotes experiments as it was a smaller dataset compared to FIGER. Finally, all experiments in zero-shot object classification have a weight decay of 5e-04.

The language tasks jointly train the parameters of the example encoder in our experiments, whereas, object classification experiments use a pretrained ResNet50 as the example encoder. They use a biLSTM with attention-based architecture as the example encoder. Furthermore, we assume a low-rank for the compatibility matrix $\mathbf{W}$. The matrix $W \in \mathbb{R}^{d \times d}$ is factorized into $\mathbf{A} \in \mathbb{R}^{h \times d}$ and $\mathbf{B} \in \mathbb{R}^{d \times h}$ where h is the low-rank dimension. Table 8 summarizes the hyperparameters used in the example encoders. Additionally, AttentiveNER has two other hyperparameters - v_m and v_f. v_m uses a pretrained 300-dim GloVe embedding. v_f learns a 60-dim feature vector during training. Lastly, in both intent classification and fine-grained entity typing, out-of-vocabulary words (oov) or words that do not have embeddings in GloVe are initialized randomly.

Task	*Inp. dim.*	*Hidden dim.*	*Attn. dim.*	*Low-rank dim.*
Intent classification	300	32	20	16
Fine-grained entity typing	300	100	100	20

Table 8: Hyperparameters for the biLSTM example encoder in the language related tasks

[1]https://allennlp.org/

In fine-grained entity typing, we have two baselines that do not use graph neural networks: OTyper and DZET. OTyper averages the GloVe embedding of 300-dim for the class representations. DZET uses a biLSTM with attention to learn the class representations. The architecture is the same as the biLSTM used to encode the context of the mention except without a window size. The input words are represented with 300-dim GloVe embeddings, which are passed to the biLSTM which has a hidden state dimension of 100. The hidden states from both the directions are concatenated to get a 200-dim hidden state vector, which is passed to an attention module. The attention module is a multilayer perceptron which learns two weight matrices of the dimension, $\mathbf{W}_h \in \mathbb{R}^{200\times 100}$ and $\mathbf{W}_a \in \mathbb{R}^{100\times 1}$. The hidden states are multiplied with the scalar attention values to get the class representation.

D.2 Graph Aggregator Summary

Task	*layer-1*	*layer-2*
Intent classification	64	64
Fine-grained entity typing	128	128
Object classification	2048	2049

Table 9: Output dimensions of the graph neural networks in our experiments

Table 9 describes the output dimensions of the node embeddings after each graph neural network layer. GCN, DGP, GCNZ, and SGN are linear aggregators and learn only one weight matrix in each of the layers. GAT learns a weight matrix for the attention where $\mathbf{W}_a \in \mathbb{R}^{2\mathrm{d}_{(1)}\times 1}$ and uses LeakyReLU activation in the attention. LeakyReLU has a negative slope of 0.2. RGCN learns B bases weight vectors in the baseline. We found that $B = 1$ performs the best for fine-grained entity typing and object classification. For intent classification, we use 10 bases, i.e., $B = 10$. In intent classification and fine-grained entity typing, the non-linear activation function after the graph neural network layer is ReLU and in object classification the activation function is LeakyReLU with a negative slope of 0.2.

ZSL-KG-LSTM learns weight matrices inside the aggregator. In the LSTM, the dimension of the hidden state is the same as the input dimension. ZSL-KG-Tr is a complex architecture with numerous parameters. In our transformer module, there are five hyperparameters - input dimension, output dimension, feedforward layer hidden dimension, projection dimension, and the number of transformer layers. We use only one transformer layer in all experiments. The input dimension and output dimensions are the same in the aggregator. For instance, in intent classification, the first layer feature for a node is a 300-dim vector and the output dim is also a 300-dim vector. For the projection dimension and the feedforward hidden layer, we just take the half times the input dimension. For instance, in intent classification, the first layer projection dimension and the feedforward hidden layer dimension is 150, i.e. half times 300.

We simulate random walks for each node in the ConceptNet graphs and compute the hit probability for the nodes in the neighbourhood. The number of steps in the random walk is 20 and the number of restarts is 10. Finally, we add one smoothing to the visit counts and normalize the counts for the neighbouring nodes.