

Explainable Zero-shot Learning via Attentive Graph Convolutional Network and Knowledge Graphs

Yuxia Geng^a, Jiaoyan Chen^b, Zhiquan Ye^a, Wei Zhang^c and Huajun Chen^a

^a College of Computer Science and Technology, Zhejiang University, Hangzhou, China

E-mails: gengyx@zju.edu.cn, yezq@zju.edu.cn, huajunsir@zju.edu.cn

^b Department of Computer Science, University of Oxford, Oxford, UK

E-mail: jiaoyan.chen@cs.ox.ac.uk

^c Alibaba Group, Hangzhou, China

E-mail: lantu.zw@alibaba-inc.com

Abstract. Zero-shot learning (ZSL) which aims to deal with new classes that have never appeared in the training data (i.e., unseen classes) has attracted massive research interests recently. Transferring of deep features learned from training classes (i.e., seen classes) are often used, but most current methods are black-box models without any explanations, especially to people without artificial intelligence expertise. In this paper, we focus on explainable ZSL, and present a knowledge graph (KG) based framework that can explain the feature transferring in ZSL in a human understandable manner. The framework has two modules: an attentive ZSL learner and an explanation generator. The former utilizes an Attentive Graph Convolutional Network (AGCN) to match inter-class relationship with the transferability of deep features (i.e., map *class knowledge* from WordNet into classifier) and learn unseen classifiers so as to predict the samples of unseen classes, with impressive (important) seen classifiers detected, while the latter generates human-understandable explanations of the transferability with *class knowledge* that are enriched by external KGs, including a domain-specific Attribute Graph and DBpedia. We evaluate our method on two benchmarks for animal recognition. Augmented by class knowledge from KGs, our framework makes high quality explanations for ZSL transferability, and at the same time improves the recognition accuracy.

Keywords: Zero-shot Learning, Knowledge Graph, Explainable AI, Knowledge-based Learning, Graph Convolutional Network

1. Introduction

Recently, object recognition by deep learning which learns features from abundant samples has gained a lot of successes. For example, it even outperforms human beings on the ImageNet ILSVRC challenges [1]. However, it still suffers from challenges from data collection: when a new class emerges, hundreds of samples are needed for training while their labels are usually hard to acquire. This makes the recognition model less competitive. Therefore, the interest in zero-shot learning is growing rapidly. It focuses on developing deep learning models for those emerging classes without training samples.

Zero-shot learning (ZSL) is widely introduced in image recognition task (e.g., [2]). It predicts the images of new classes (i.e., unseen classes) that do not exist in the training set by transferring features learned from the training classes (i.e., seen classes). The inspiration is that *human can recognize new objects through the class knowledge (e.g., description) itself, even with no labeled samples*. For example, considering the animal class “*Serval*”, even though a human being might have never seen samples in the past, s/he would still be able to recognize it based on the description: “*Serval*, a kind of animal with a *Cat*-like face and a *Cheetah*-like body” (see Figure 1). With previous recognition experience of *Cat* and *Cheetah*, s/he can easily reason

about the specific appearance of *Serval* and identify it correctly.

The general principle of most ZSL algorithms is to represent such class knowledge and utilize inter-class relationship to transfer model parameters such as neural network features from seen classes to unseen classes. Some works (e.g., [3, 4]) leverage the embedding of class names learned from text corpus for transferring e.g., CNN features, while others (e.g., [5, 6]) prefer to more complex knowledge like class hierarchy and class attributes. These methods aim at learning and prediction for unseen classes ([3–8]), but are black-box models: the transferability of features is uninterpretable. This not only limits human’s trust on the prediction, but also disables the human-machine interaction which is important in machine learning model developing, configuration and debugging.

There have been few works that explain ZSL with human understandable knowledge. As far as we know, the only work that is close to ours is by Selvaraju et al. [9]. They first learned the mapping between class attributes and neuron importance, and then transferred neurons (i.e., features) from seen classes to unseen classes, where attributes of classes were used to justify the prediction of unseen classes (cf. more in Section 2). Such work indicates that it’s feasible to explain ZSL by class knowledge such as class attributes. However, this work focuses on explaining the neuron importance, but ignores the feature transferability which is the core of ZSL. Moreover, its method is ad-hoc, only working for predefined class attributes, while our explanation method supports not only attributes but also general common sense knowledge in different formats, coming from external KGs like DBpedia.

In this paper, we propose a KG based framework to explain the feature transferability in ZSL. It first adopts a KG named WordNet and an Attentive Graph Convolutional Neural Network (AGCN) to model inter-class relationship for ZSL, which is also known as an *Attentive ZSL Learner* (AZSL). Namely, a matching between the inter-class relationship and the transferability of CNN features from seen classes to unseen classes is learned. It then uses an *explanation generator* to extract rich class knowledge from both domain-specific Attribute Graph and general external KGs (e.g., DBpedia) as common sense evidences for ZSL explanation. For example, considering the ZSL case in Figure 1, the attribute knowledge of *sharp ear* and the DBpedia knowledge of *felidae ancestor* can explain the positive feature transferability from *Cat* and *Cheetah* to *Serval*. Finally, we propose several

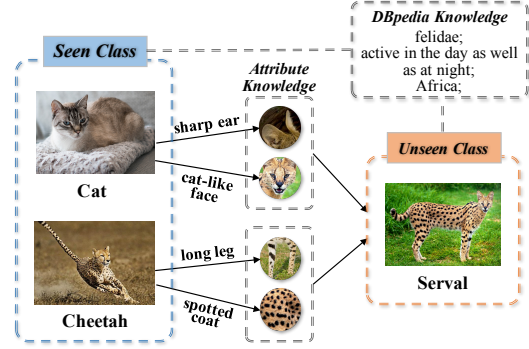


Fig. 1. An example of recognizing *Serval* (unseen class) with two seen classes (*Cat* and *Cheetah*). We focus on explainable ZSL, which captures domain-specific attributes and general common sense, such as sharp ear, face appearance, long leg, spotted coat and felidae ancestor, as evidences to make explanations that can be understood by humans without artificial intelligence expertise.

templates to generate human understandable explanations.

Briefly, our work contributes in the following aspects:

- A KG-based explanation framework for zero shot learning is proposed. It is among the first to explain the transferability of neural network features in ZSL.
- A novel ZSL algorithm called AZSL is built upon WordNet and AGCN. It models the inter-class relationship and the transferability of CNN features from seen classes to unseen classes, which not only shows improvements over the state-of-the-art baselines, but also enables explaining the transferability of CNN features in ZSL.
- An explanation generator is developed. It can generate ZSL explanations with class semantics from not only domain-specific Attribute Graph but also general KGs like DBpedia.
- A set of templates are designed to re-organize the transferability explanations and generate human-consumable natural language descriptions for ZSL explanation.
- Lastly, extensive experiments are conducted to evaluate the generated explanation and the ZSL learner, using two image classification benchmarks. The explanation achieves high quality according to the analysis on different metrics and human assessment.

The structure of this paper is as follows. In Section 2, we review the related work. In Section 3, we set up the background of our work. In Section 4, we introduce

the details of our KG-based explanation framework, including the attentive ZSL learner in Section 4.2 and the explanation generator in Section 4.3. In Section 5, we display the experiments and the evaluation. Finally we conclude the paper and present the future work.

2. Related Work

2.1. Zero-shot learning

Zero-shot learning (ZSL) has received a lot of attention in machine learning community. Some work by Larochelle et al. [10] has shown the ability to predict new (unseen) classes of digits that were omitted from the training set, with the features from training (seen) classes being transferred. In computer vision, techniques for utilizing the knowledge of classes to realize the transfer of deep features from seen classes to unseen classes have been investigated [2, 3, 5, 10, 11].

Early algorithms focus on utilizing class attributes to model the semantic relationship of classes [6, 12–14]. For instance, Lampert et al. [6] annotated each class with an attribute representation and proposed two methods for attribute-based classification, where the features can be transferred between seen and unseen classes via attribute sharing. Recent methods prefer to utilize class embeddings trained with class text description to explore the class semantics and relationship [3, 4, 15, 16]. For example, Frome et al. [3] presented a *visual-semantic embedding* model, which leverages textual data to model the semantic relationships between classes, and linearly maps image features into the semantic embedding space for feature transferring. However, the state-of-the-art performance in ZSL image recognition is achieved by those who utilize KGs for class relationship [5, 17, 18]. For example, Wang et al. [5] proposed a GCN-based transfer method and used WordNet to model the relationship of hierarchical classes. In our paper, we combine class embeddings and class hierarchy to constitute the knowledge of classes, and utilize Attentive GCN to encode this class knowledge for feature learning in ZSL.

There are also some ZSL methods for dealing with the sample shortage challenge in other domains, especially in natural language processing (NLP) tasks including text classification [19, 20], entity linking [21, 22], relation extraction [23] and others [24]. These methods also work on introducing high-level resources about labels as label knowledge to build the relationship of seen and unseen labels. Moreover, the sample

features of NLP data are closer to label knowledge due to the same symbolic representation, which is beneficial for feature transferring in NLP domain. While the feature transfer learning in our work is more challenging considering the gap of visual and symbol field.

2.2. Explainable Artificial Intelligence

Explainable artificial intelligence (AI), which aims to produce interpretable models or predictions, is becoming more and more popular nowadays [25–28]. Such methods enable human beings to understand, trust and effectively manage the AI systems and their decisions. Some of the explanation works design white-box and inherently interpretable models like rule-based systems [29], while others try to justify the prediction of a black-box model by for example approximating its behaviour locally with simple interpretable linear models [30], or quantifying the contribution of each single input variable [31].

Most explanations target at humans with AI expertise. However, there is also a great need for common people without AI expertise such as medical doctors to understand the decisions made by AI-based systems [32]. Most of these works utilize natural language corpora to generate human readable explanations. Studies like Biran et al. [33] introduced linguistic expressions from Wikipedia articles to explain stock price prediction to common people. Li et al. [34] proposed to generate attributes and captions of images as explanations to indicate whether the system really understand the image content when answering a visual question.

There are also a few works devoting to enrich the explanation with knowledge graphs (KGs), by utilizing human understandable background and common sense knowledge in these KGs, as well as their underlying semantics that can be inferred by reasoning [35, 36]. Tididi et al. [36] presented a framework which exploited Link Data as background knowledge to generate explanations for data clusters. Chen et al. [37] utilized Semantic Web techniques to extract human understandable evidences from local domain ontologies and external KGs like DBpedia to explain the results of flight delay forecasting.

Another related research direction is utilizing the attention mechanism to make explanations [38, 39]. For example, Yang et al. [38] utilized attention layers to illustrate that the model selected qualitatively informative words and sentences in document classification. Our work utilizes such attention techniques for ZSL, but goes beyond the attention. It includes a general

framework to incorporate semantics from KGs, and generates human-centric explanations for the core of ZSL – deep feature transfer.

2.3. Transfer Learning Explanation

ZSL is often regarded as a branch of transfer learning which aims at utilizing samples, features or model parameters learned from one domain to guide the learning in another domain [40, 41]. ZSL algorithms usually transfer features learned deep neural networks from seen classes (domains with labeled training samples) to predict testing samples of unseen classes (domains without labeled training samples).

Some works have been proposed to augment transfer learning as well as ZSL with KGs [37, 42–44]. For example, in [44], prior knowledge about the prediction task and domains were expressed by ontologies, and further utilized to analyze the transferability of features and samples for augmentation. For another example, Zhang et al. [42] proposed a transfer learning based algorithms for long-tail relation extraction, which incorporated the data features from data-rich relations for tackling the prediction of data-poor relations. Knowledge of the relation, which comes from from a KG, was investigated to enhance the feature learning for data-poor relations, using KG embeddings and relation hierarchy. In summary, these works indicate the feasibility of studying transfer learning tasks and domains by external knowledge from KGs. In our ZSL study, we not only utilize KGs for performance improvement (i.e., the attentive ZSL learner based on KG and AGCN), but also for human understandable explanations.

Recent studies on transfer learning explanation focus on the analysis of feature transferability [37, 45–47]. For example, Liu et al. [46] assumed that the feature is transferable from a source domain to a target domain if the source and target domains have some similar feature structures. For another example, Chen et al. [37] extracted knowledge (ontology axioms and DBpedia facts) that co-exist in the source and target domain to explain the transferability of features learned by deep neural network. These works indicate that the transferability of features is highly correlated with the knowledge of the source and target domain. Our work in this paper also prefers to extract domain knowledge for feature transferability explanation. Different from the above works, it on one hand develops a general framework that can generate explanations from different knowledge from multiple KGs such as the domain-

specific Attribute Graph and DBpedia. On the other hand, it focuses KG-based ZSL – an important and popular transfer learning branch whose current solutions are all black-box models without explanations.

Few works have been found to explain ZSL with KGs. The only work we know is Selvaraju et al. [9]. It first learned a mapping between class attributes and the importance of individual neurons, and then predicted unseen neuron importances by semantically composing those of seen classes to optimize unseen classifiers. The inverse mapping of transferred neuron importances (i.e., attributes of classes) were taken as explanations to validate the decisions made by classifiers. In this work, the authors focused on explaining the prediction of unseen classes, however, ignored the feature transferability in ZSL. Contrastingly, the explainable ZSL proposed in our work pays attentions to the transferability of features, which is more appropriate to explaining the nature of ZSL. At the same time, the explanations we generate contains not only domain-specific attributes but also general common sense knowledge, which is more expressive and flexible compared with the ad-hoc class attributes in [9].

3. Preliminaries

3.1. Zero-shot Learning

In zero-shot learning, the training set is denoted as $\mathcal{D}_{tr} = \{(x_i, l_i)\}_{i=1}^N$, where N is the number of training samples, x_i represents the i -th training image and l_i is its label. While the testing set is described as $\mathcal{D}_{te} = \{(\tilde{x}_i, \tilde{l}_i)\}_{i=1}^{\tilde{N}}$, and its labels have no overlap with the labels in \mathcal{D}_{tr} . We regard the labels in \mathcal{D}_{tr} as *seen classes*, denoted as S , and the labels in \mathcal{D}_{te} as *unseen classes*, denoted as U . Each class involves an unique classifier f for predicting whether a sample is of the class or not. ZSL aims to learn classifiers for unseen classes and predict the labels for testing samples in \mathcal{D}_{te} by transferring features (sample representations) learned from \mathcal{D}_{tr} .

3.2. Class Knowledge

In our study, we introduce three kinds of Knowledge Graphs (KGs) to depict the *class knowledge*, which describes the semantic relationship between classes. They are used for feature transfer in ZSL as well as generating its explanations. We briefly introduce the three KGs below.

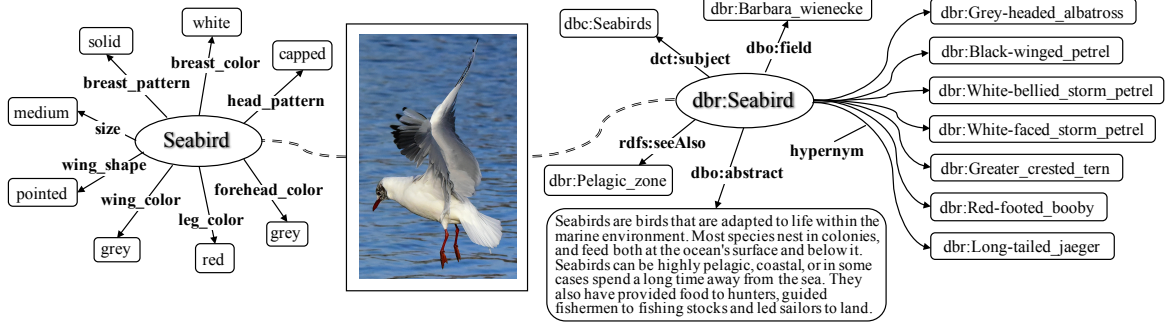


Fig. 2. An example of our two introduced KG resources for animal class *Seabird*. [Left] the domain-specific Attribute Graph with corresponding entity *Seabird*; [Right] the general DBpedia with aligned entity *dbr:Seabird*.

WordNet [48] is a lexical knowledge base for English where nouns, verbs, adjectives and adverbs are organized into sets of synonyms, each representing a lexicalized entity. Semantic relations (hypernym, hyponymy, meronymy, etc.) are used to link these entities. We utilize such a KG to build a hierarchical structure of classes, by aligning each class with an entity in WordNet. In this structure, the edge that connects two class nodes represents the “subClassOf” relationship.

Attribute Graph is a domain-specific knowledge graph we created by collecting attribute annotations that describe the visual characteristics of objects. The knowledge about one class from such a special KG is in the form of *key-value*, where *key* represents the attribute item of object and *value* is the corresponding attribute value. We organize these *key-value* knowledge in the format of triple (o, a, v) , where o represents a visual object class, a and v represent *key* and *value* respectively. Taking *Seabird* as an example, as Figure 2 [Left] shows, we can describe its attribute annotation “grey wing color” by the triple $(Seabird, wing_color, grey)$. According to studies in computer vision [12, 49–52], such attributes of many data sets are now available, while in a specific real world application, they can come from domain knowledge or experts.

DBpedia [53] is a general knowledge graph that includes common sense and background knowledge about classes. With knowledge from Wikipedia encyclopedia, DBpedia is a large scale KG consisting of 4.58 million entities and 3 billion facts. The object classes in ZSL can be matched to entities of DBpedia. For example, the class *Seabird* in Figure 2 [Right] can be matched to the entity *dbr:Seabird*. Different from Attribute Graph, DBpedia contains general knowledge, such as the background description from property *dbo:abstract* and the common sense relation from *hypernym* entities.

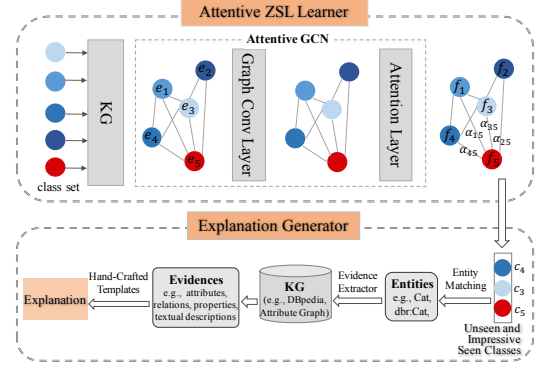


Fig. 3. Our proposed KG-based explainable ZSL framework.

4. Methodology

4.1. Framework Overview

In this paper, we present a KG-based framework to explain the feature transferability in ZSL in a human understandable manner, including an Attentive ZSL learner (AZSL) and an explanation generator (EG), as illustrated in Figure 3. AZSL first models the hierarchy relationship of seen classes, unseen classes as well as their ancestor and descendent classes using WordNet, and then maps this class knowledge into the deep features learned by CNNs, which pursue class discrimination and are core components of a classifier. Instead of simple linear mapping, we utilize a GCN to encode the inter-class relationship and non-linearly map features of seen classes to unseen classes. Considering the different contributions of features from different seen classes, we extend the GCN with an additional attention layer to learn attention weights of seen classes. Those weights provide an elementary illustration of the relationship between seen and unseen classifiers (i.e., the transferability of deep features). Based

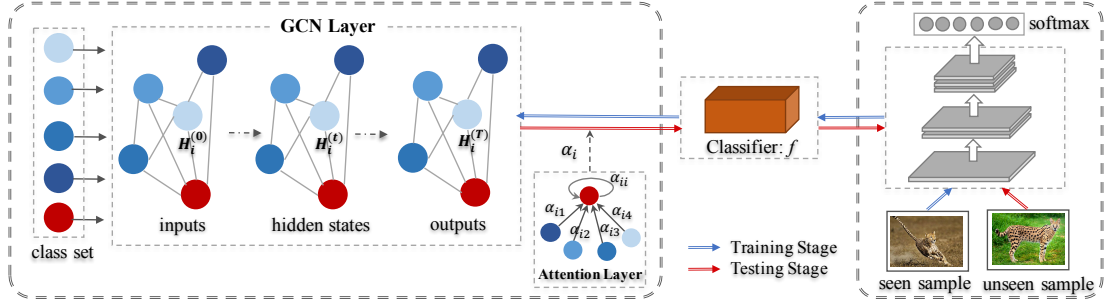


Fig. 4. An overview of our attentive ZSL learner. At training stage, AZSL maps class knowledge into classifiers with GCN and an additional attention layer; At testing stage, AZSL predicts classifiers for unseen classes and conduct nearest searching to classify the samples of them.

on this, EG then matches both seen and unseen classes with entities of external KGs, from which richer class knowledge, such as entity attributes, relations, properties and textual descriptions, are extracted as evidences, to make a further illustration of the inter-class relationship and explain the transferability of features. It also generates natural language explanations with these evidences using some hand-crafted templates.

4.2. Attentive ZSL Learner

AZSL matches inter-class relationship with the transferability of deep features to learn classifiers of unseen classes via an Attentive Graph Convolutional Network (AGCN). As Figure 4 shows, AZSL includes: (i) learning a discriminative *classifier* for each class (Figure 4 [Right]); (ii) encoding class knowledge into classifiers (Figure 4 [Left]); (iii) predicting classifiers for unseen classes.

4.2.1. CNN Classifier

Consider Convolutional Neural Network (CNN), a frequently used model for feature extraction in object recognition, which predicts label l_i for image x_i . This deep networks contain some layers for feature learning, where the significant features of images are extracted to make predictions. Especially the output features of the second to the last layer pursue class discrimination when trained with samples of different classes. We take these features as classifier to represent the deep features for each class.

Given a class c , the classifier f_c is learned by training CNN with its samples $\{x_i^c | i \in n\}$, as shown in Figure 4 [Right]. In our model, we pre-train classifiers for seen classes with samples from \mathcal{D}_{tr} , and use them to train the mapping of class knowledge in next section. After doing this, we predict classifiers for unseen classes to recognize the samples from \mathcal{D}_{te} .

4.2.2. Mapping Class Knowledge into Classifier

We utilize Graph Convolutional Network (GCN) to encode the graph-structured inter-class relationship.

Graph Convolutional Network works on capturing the dependence of graph-structured data via information propagation between the nodes in the graph. In each layer, convolutional operator learns a node's representation by aggregating features from its surrounding nodes defined in the graph, and propagates them to the next layer. Mathematically, we describe the propagation of each layer as:

$$H^{(t+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(t)} W^{(t)}) \quad (1)$$

The input of t -th layer $H^{(t)} \in \mathbb{R}^{M \times h}$ is a series of hidden features of nodes, where M is the number of nodes, h is the dimension of feature representation for each nodes. And $H^{(t+1)}$ is the updated features of nodes, as the output. The convolutional operator is supported by the eigendecomposition of adjacency matrix A and degree matrix D of the graph with M nodes. $\tilde{A} = A + I_M$ is A with added self-connections, I_M is the identity matrix, and \tilde{D} is the diagonal matrix such that $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $W^{(t)}$ represents a layer-specific trainable weight matrix, and $\sigma(\cdot)$ is an activation function like LeakyReLU with negative input slope 0.2 (More details are in [54]).

In our ZSL context, the input of first layer ($H^{(0)}$) is the initial representations of class nodes. Motivated by [3, 4], we choose pre-trained word embeddings of class names as the initial features. These *embeddings*, learned from a large scale corpus ([55]), are semantically meaningful representations of the classes. Therefore, given classes from seen set S and their initial representations $e_{1...|S|}$, we learn a GCN model, denoted as \mathcal{M}_{GCN} . The output of the last layer of \mathcal{M}_{GCN} is the predicted classifiers $\hat{f}_{1...|S|}$. With ground-truth classi-

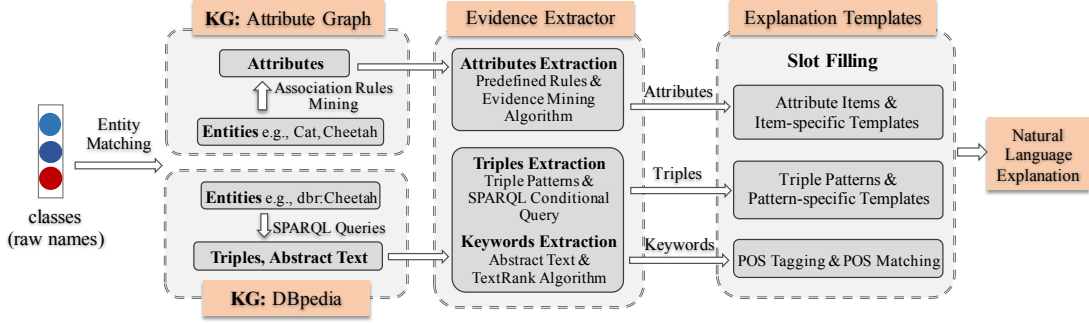


Fig. 5. Illustrating for generating natural language explanation from external Attribute Graph and DBpedia.

fiers $f_{1...|S|}$, we use mean square error as loss function to train the parameters of \mathcal{M}_{GCN} :

$$\text{loss} = \frac{1}{|S|} \sum_{i=1}^{|S|} \mathcal{L}_{MSE}(f_i, \hat{f}_i) \quad (2)$$

$$\hat{f}_i = \mathcal{M}_{GCN}(e_i)$$

4.2.3. Predicting Unseen Classifier

Based on the feature propagation in \mathcal{M}_{GCN} , the classifier of an unseen class can be learned by propagating features from its surrounding seen classifiers, in this way, the deep features are transferred from seen classes to unseen classes. Therefore, we predict classifier for unseen class u with its class embedding e_u as

$$\hat{f}_u = \mathcal{M}_{GCN}(e_u) \quad (3)$$

Moreover, we find that different surrounding classes have different influence towards the learning of a specific class's classifier in the feature transferring. Hence, we utilize the attention mechanism — stacking an attention layer after GCN and retraining to model different contributions of different surrounding classes.

For a certain class i , we assign different attention weights to its surrounding classes by computing the similarity of the classifier of i and its surroundings, because when a surrounding contributes more to i in the transferring, their learned classifiers are more similar. As Figure 4 [Left] shows, the contribution of a surrounding class j to class i is computed as:

$$\alpha_{ij} = \frac{\exp(\cos(\hat{f}_i, \hat{f}_j))}{\sum_{k \in \mathcal{N}_i} \exp(\cos(\hat{f}_i, \hat{f}_k))} \quad (4)$$

where $\cos(\cdot)$ denotes the cosine similarity, \mathcal{N}_i denotes the set of surrounding classes of class i , including class

i itself. The computed attention weights are used to update the mapped classifier of class i as:

$$\hat{f}_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij} \cdot \hat{f}_j \quad (5)$$

We define a surrounding class of a specific class as an **impressive class** if its attention weight exceeds a threshold α , and the corresponding classifier is taken as **impressive classifier**. As for each unseen class in the set U , we get its **impressive seen classifiers**, each of which is believed to be important in transferring features towards the unseen classifier. By this means, we elementarily make the relationship between seen and unseen classifiers transparent, which are the base to explain the feature transferability in ZSL.

At test time, when a new image arrives, AZSL first extracts its deep features using pre-trained CNN, and then looks for the nearest predicted unseen classifiers. The nearest classifiers are then mapped back to unseen classes for scoring so that predicting label for it.

4.3. Explanation Generator

With the learned relationship between seen and unseen classifiers, we introduce two external knowledge graphs, domain-specific Attribute Graph and general DBpedia, to extract reliable evidences and generate human understandable explanations for justifying the transferability of features.

The explanation generating procedure is illustrated in Figure 5. Briefly, we (i) match raw class names with entities of external KGs; (ii) extract supported evidences using different strategies for different external KGs; and (iii) generate natural language explanations with some templates.

4.3.1. Domain-specific KG: Attribute Graph

Given an unseen class and its impressive seen class set, we desire for extracting common attributes from Attribute Graph as evidences to validate the feature transferability between them. However, the searching space is often large for finding common attribute set, especially when multiple impressive seen classes exist. To this end, we develop a rule-mining based method to find out the common attributes by mining association rules of classes from Attribute Graph, with an algorithm named *EvidenceMining*. The mined rules illustrate the semantic association between classes, which can validate the rationality of the learned feature transferability relationship between them. The supporting of a rule is the common attributes shared by these classes, which are desired evidences to explain the feature transferability from the seen classes to the unseen classes.

Association rule mining is a widely used algorithm in Data Mining. It was first proposed for mining the association rules of items from a list of customer transactions [56]. Each transaction consists of a set of items purchased by a customer in a visit. An association rule of items is like $\{bread \Rightarrow milk\}$, meaning that people who purchase *bread* usually also purchase *milk*. In the context of mining association rules of classes, a transaction is defined as an attribute is both owned by a set of classes. The rule of classes means that these classes are associated because they share a set of attributes.

Let $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$ and $\mathcal{C} = \{c_1, c_2, \dots, c_m\}$ be the set of attributes and the set of classes of Attribute Graph \mathcal{G} respectively. Let \mathcal{D} be a set of transactions, where each transaction is labeled with a_i and consists of a set of classes \mathcal{C} both having attribute a_i . An association rule is an implication of the form $\{X \Rightarrow Y\}$, where X and Y are sets of classes, $X \subset \mathcal{C}$, $Y \subset \mathcal{C}$, and $X \cap Y = \emptyset$. The rule $\{X \Rightarrow Y\}$ holds in the transaction set \mathcal{D} when $c\%$ of attributes in \mathcal{D} owned by X are also owned by Y ; here $c\%$ is called the *confidence* of the rule. The *support* of the rule $\{X \Rightarrow Y\}$ is the ratio of attributes that are owned by both X and Y . These support attributes are common attributes of the classes in $X \cup Y$. Regarding the unseen class u and its related seen class set $S = \{s_1, \dots, s_n\}$, the potential association rule of them can be predefined as

$$\begin{aligned} &\{s_1\} \Rightarrow \{u\} \\ &\dots \\ &\{s_n\} \Rightarrow \{u\} \\ &\{s_1, \dots, s_n\} \Rightarrow \{u\} \end{aligned}$$

Table 1

Example of mining association rules of classes *polar bear*, *raccoon* and *grizzly bear*.

(a). Database \mathcal{D}

Transaction Label	Class Items
claws	polar bear, raccoon, grizzly bear
black	raccoon, grizzly bear
furry	raccoon, grizzly bear, polar bear

(b). Frequent Classsets

Classset	Support Attributes
$\{polar\ bear\}$	claws, furry
$\{raccoon\}$	claws, black, furry
$\{grizzly\ bear\}$	claws, black, furry
$\{polar\ bear, grizzly\ bear\}$	claws, furry
$\{raccoon, grizzly\ bear\}$	claws, black, furry
$\{polar\ bear, raccoon\}$	claws, furry
$\{polar\ bear, raccoon, grizzly\ bear\}$	claws, furry

(c). Rules

Rule	Support	Confidence
$\{polar\ bear\} \Rightarrow \{grizzly\ bear\}$	66.6%	100%
$\{raccoon\} \Rightarrow \{grizzly\ bear\}$	100%	100%
$\{polar\ bear, raccoon\} \Rightarrow \{grizzly\ bear\}$	66.6%	100%

, the support sets of these rules are common attributes of the classes in these rules.

Take unseen class *grizzly bear* and its impressive seen classes *polar bear* and *raccoon* as an example. Let $\mathcal{C} = \{polar\ bear, raccoon, grizzly\ bear\}$ and $\mathcal{A} = \{claws, black, furry\}$. Consider the transaction database \mathcal{D} shown in Table 1. The sets of classes with pre-specified minimum support (i.e., frequent classsets) and the rules corresponding to these classsets are shown in Table 1. We can mine the association rule “ $\{polar\ bear, raccoon\} \Rightarrow \{grizzly\ bear\}$ ”, as well as its supporting attributes: $\{claws, furry\}$, which are common attributes shared by the three classes and can be taken as evidences for validating the transferability from *polar bear* and *raccoon* to *grizzly bear*.

Algorithm 1 illustrates the mining pseudocode of rules and evidences. Given an unseen class and its impressive seen classes, we first instance the predefined rules and extract annotated attributes of each class from Attribute Graph \mathcal{G} to construct the transaction database as shown in Table 1(a). Then, we apply Apriori algorithm [57] to mine the frequent classsets with pre-specified minimum support value. Constraining with predefined rules and minimum confidence value, we filter frequent class sets to get the association rules, and finally count the rules’ support attributes in \mathcal{D} as evidences to output.

Thus we not only mine association rules of seen and unseen classes with measurements e.g., support and

Algorithm 1 Evidence Mining

Input: Attribute Graph \mathcal{G} ; Unseen class u and its impressive seen class set S ; Minimum support and confidence value s_{min}, c_{min} ; Predefined rule set R ;

Output: A : explanatory evidences for u and S ;

```

1:  $\mathcal{C} = \{u, s_1, \dots, s_n\}$ ; % Prepare class set
2:  $R_u = \text{instance}(R, \mathcal{C})$ ; % Prepare rule set for  $u$ 
3:  $\mathcal{A} = \emptyset$ ; % Init. of attribute set
4: for each class  $c \in \mathcal{C}$  do
5:   % Get attributes of  $c$ 
6:    $\mathcal{A}_c = \text{ExtractAttribute}(\mathcal{G}, c)$ ;
7:   Append  $\mathcal{A}_c$  to  $\mathcal{A}$ ;
8: end for
9:  $\mathcal{D} = \text{ConstructDataset}(\mathcal{G}, \mathcal{C}, \mathcal{A})$ ; % Store Trans.
10:  $k = n + 1$ ; % Size of frequent class set
11: % Get frequent class set and support value
12:  $F, v_s = \text{Apriori}(\mathcal{D}, k, s_{min})$ ;
13: % Filtering for confident rule set
14:  $R'_u, v_c = \text{Filter}(F, R_u, c_{min})$ ;
15: % Count support attributes as evidences
16:  $A = \text{count}(\mathcal{D}, R'_u)$ ;
17: return  $A$ ;
```

confidence, but also extract common attributes from Attribute Graph as evidences to explain the transferability between classes.

4.3.2. General KG: DBpedia

Different from Attribute Graph whose entities can be exactly aligned with the ZSL classes by name, the matching between DBpedia entities and ZSL classes are more challenging due to the ambiguity. One widely used and effective approach is lexical matching, with an index on the entity's name, label, anchor text (description), etc. In our paper, we use the DBpedia Lookup service¹, which is based on the index of DBpedia Spotlight [58]. Specifically, we use raw class names as keywords to look up the corresponding DBpedia entities. For example, the entity “*dbr:Cheetah*” can be looked up by the name string “*Cheetah*”.²

One challenge of class to entity matching is that only a part of all the classes have entity correspondences. On one hand, the corresponding entity of the class does not exist in the KG. For example, DBpedia only has an entity for *Chicken* but no *Cock* and *Hen*. The latter two however have totally different visual features. On the other hand, some classes are

wrongly matched to entities. This is some degree can be solved by filtering out those retrieved entities that are instances of wrong types w.r.t. the domain of the ZSL problem. For example, *Red fox* is incorrectly matched with entity *dbr:Fox*, while the correct matching should be *dbr:Red fox*. We remove it from the match set.

Different from the fixed attribute annotations of classes in Attribute Graph, the knowledge about classes in DBpedia is massive and diversified. Thus, with matched entities, we utilize SPARQL queries³ to retrieve two kinds of evidences: 1) **abstract text** which is an overall description of entity with keywords included, and 2) structured **triples** which describe fine-grained semantics of an entity, e.g., properties and relations with other entities.

Two kinds of triples are extracted: (i) object triple, denoted as (h, r, t) , where h is the head entity, t is the tail entity, and r is the relation; (ii) property triple, denoted as (h, p, v) , where h is the head entity, p is the data property and v is the data value (literal). With these triples, we can find some items associating both seen and unseen entities to illustrate their common knowledge. However, only a portion of all the triples of an entity are useful for describing the common knowledge. Thus we need a method to find out those triples efficiently.

To this end, we develop a strategy based on heuristically triple pattern for extraction. Some *triple patterns* are designed, as shown in Table 2. Based on these patterns, we generate corresponding SPARQL queries to retrieve the common knowledge between entities (classes), including shared relations and properties. Considering the example in Figure 1, *Cat* and *Serval* share the same ancestors *Felidae*. This fact can be verified by two triples (*dbr:Cat*, *hypernym*, *dbr:Felidae*) and (*dbr:Serval*, *hypernym*, *dbr:Felidae*), both of which are extracted according to the pattern $((s, r_1, t) \wedge (u, r_2, t))$. Note that some extracted triples may be not accurate enough. For example, the object entity *dbc:Birds_of_Europe* in the triple (*dbr:Ruddy_Turnstone*, *dbo:family*, *dbc:Birds_of_Europe*) is too general with regard to bird species in image recognition of birds. This in some degree brings useless information, and is a factor that impacts the quality of explanations.

Apart from triple annotations, each entity of DBpedia has a textual description with a set of sentences (i.e., abstract text), which is annotated by the prop-






¹<https://github.com/DBpedia/lookup>

²*dbr*, *dbo*, etc. are URI prefixes in DBpedia. Please see <http://DBpedia.org/sparql?help=nsdecl>.

³<https://www.w3.org/TR/rdf-sparql-query/>

Table 2

Triple patterns and corresponding SPARQL query items, where s, u represent entity patterns corresponding to seen and unseen entities respectively, \wedge represents the joint operator of patterns.

Triple Pattern	Diagram	Query Item	Illustration
(s, r, u)		SELECT ?r WHERE { s ?r u. }	s is directly related with u via a specific relation r .
(u, r, s)		SELECT ?r WHERE { u ?r s. }	u is directly related with s via a specific relation r .
$(s, r_1, t) \wedge (u, r_2, t)$		SELECT ?r1 ?r2 ?t WHERE { s ?r1 ?t. u ?r2 ?t. }	s and u both connect with another entity t via respective relation r_1 and r_2 (r_1, r_2 may refer to the same relation).
$(s, p, v) \wedge (u, p, v)$		SELECT ?p ?v WHERE { s ?p ?v. u ?p ?v. }	s and u both have property p and share the same property value v .
$(s, r_1, t) \wedge (t, r_2, u)$		SELECT ?r1 ?r2 ?t WHERE { s ?r1 ?t. ?t ?r2 u. }	s and u is related via a transitional entity t .

erty of *dbo:abstract* and thus can be accessed by a SPARQL query. Through the abstract text, we can extract informative descriptions of class, especially those about visual characteristics. Whereas, some sentences in abstract text are general descriptions: e.g., a sentence, “Dogs perform many roles for people, such as hunting, herding, protection, assisting police and military, companionship and, more recently, aiding handicapped individuals”, describes the social background of *dbr:Dog*, words and phrases in this sentence do not mention much discriminative properties about *Dog*.

To utilize the knowledge in abstract text, we adopt TextRank [59], an unsupervised automatic summarization algorithm that can extract keywords from the text. Core descriptive words and phrases are extracted for an entity. They provide class-specific properties thus contributing to the extraction of informative common properties between entities. For example, the extracted keyword *Africa* in Figure 1 expresses the same living environment of *Cheetah* and *Serval*.

4.3.3. Template-based Explanation Generator

These aforementioned common items, including attributes, triples and keywords that are extracted from Attribute Graph and DBpedia, constitute fine-grained class knowledge that can be used as evidences to explain the transferability between seen and unseen classes. In order to make them more understandable for common people without AI expertise, we feed them into hand-crafted templates to generate natural language explanations.

Inspired by *Slot Filling*, a popular method of completing information in dialogue system, we design templates with entities (classes), attributes, relations, prop-

erties and keywords as slots, and take the extracted items as values to fill in. We totally design three different kinds of templates, as shown in Table 3, for the structured triples, unstructured attributes and keywords respectively.

Attributes are domain-specific descriptions used for annotating visual objects. The attributes belonging to the same attribute item describe the same aspect of object. For example, some attributes such as *head*, *tail*, *claws*, *leg* describe the body parts of animals, which refers to the attribute item *body*, while others like *red*, *green*, *blue* describe the appearance color of objects, which refers to the attribute item *color*. Considering the attributes of the same attribute item can be expressed in similar sentences, we design templates based on different items, which are fed by attributes extracted from Attribute Graph. For example, the common attribute *sharp ear* of *Cat* and *Serval*, which refers to the attribute item *body*, can be expressed with the sentence: “They both have sharp ear”. Table 3 lists some of the items and their corresponding templates. Additionally, the number of extracted common attributes varies. In order to restrict the length of generated sentence, we randomly select 10 attributes to describe the common knowledge between seen and unseen classes when the number of common attributes exceeds 10.

Structured triples follow some common formats, especially for those extracted based on identical *triple patterns*. Thus we design templates as shown in the center of Table 3 to textualize triples according to their *triple patterns*, where entities, properties and relations of the triples are used as values to fill the corresponding slots in templates. Taking triples (*dbr:Cat*,

Table 3

Explanation templates for generating textual explanation. A overall illustration is first provided to summarize the explanation. s, u, t, r, p, a, adj , etc. are slots in templates to be filled. The Left is for attributes, where we list part of all attribute items. The Center is for structured triples. The Right is for textual keywords, in which we list all possible Part-of-Speech (POS) of keywords, including adjective (*adj*), noun (*n*) and named noun (*named n*). Notably, the POS attached to attribute item (e.g., *coat (adj)*) means the attribute values with different POSs.

Overall Illustration: The prediction for samples of u is supported by s .					
Attributes & Templates		Triples & Templates		Keywords & Templates	
Attribute Item	Template	Triple Pattern	Template	POS of Keyword	Template
<i>color, size, species, coat (adj), ...</i>	They are both a .	$(s, r, u), (u, r, s)$	$s(u)$ is r of $u(s)$.	<i>adj</i>	They are both [adj] animal/bird/other.
<i>body, shape, coat (n), ...</i>	They both have a .	$(s, r_1, t) \wedge (u, r_2, t),$ $(s, r_1, t) \wedge (t, r_2, u)$	s and u are both relevant to t via relation r_1, r_2 . (or s and u are both a member of t .)	<i>n</i>	They both have [n] . (or they are similar in [n] .)
<i>feeding, habitat</i>	They both eat (or live in) a .	$(s, p, v) \wedge (u, p, v)$	s and u share the same v of property p .	<i>adj + n</i>	s and u are similar in [adj+n] . (or s and u both have [adj+n] .)
<i>behaviour, habits</i>	They both behave (or like) a .	$(s_1, r_1, u) \wedge (s_2, r_2, u)$	s_1, s_2 both belong to u . (or s_1, s_2 are both species of u .)	<i>named n (location)</i>	They both live in [n.location] .

hypernym, dbr:Felinae) and (*dbr:Serval, hypernym, dbr:Felinae*), which are extracted for *dbr:Cheetah* and *dbr:Serval* with pattern “ $(s, r_1, t) \wedge (u, r_2, t)$ ”, as examples, the following textual explanation generated: “*Cat* and *Serval* are both relevant to *Felinae* via relation *hypernym*”. Notice that the DBpedia prefixes such as “*dbr*” in the triple will be removed when generating sentences.

Keywords extracted from the abstract text are flexible expressions consisting of adjectives, nouns, their combinations and so on. One example is *spotted coat*. Therefore, we utilize *POS Tagging* and *POS Matching* to generate sentences with keywords. Specifically, each extracted keyword is first labeled with an additional Part-of-Speech (POS) tag, and templates are designed based on these POS tags as the right of Table 3 shows. Then, the sentence is generated by matching POS-tagged keywords with corresponding POS slots. For example, one sentence generated with the common keyword *spotted coat* of *Cheetah* and *Serval* is “They are similar in spotted coat.” In particular, some nouns have specific meanings (i.e., *named nouns*), such as habitats being a location nouns. Therefore, we also use named entity recognition tools to annotate these special nouns in sentences and design the right templates.

Moreover, to generate more readable explanations with flexible expressions, we further enrich the word in templates with its synonyms from WordNet [60].

5. Evaluation

We conduct experiments on image classification and evaluate our framework in following aspects: (1) accuracy of our attentive ZSL learner (AZSL) in comparison with the state-of-the-art ZSL baselines; (2) quantitative analysis of the feature transferability; (3) evaluation on the textual explanations of transferability, including human scoring, qualitative analysis and case studies. Based on the evaluation, we also discuss the relationship between feature transferability and ZSL prediction performance.

5.1. Experiment Setting

5.1.1. Datasets

Two widely used image sets are adopted: Animals with Attributes (AwA) [6] and ImageNet [61]. In AwA, each image is annotated by attributes and each class is associated with a set of attributes. ImageNet contains over 21,000 classes but has no attributes annotated. Each ImageNet class corresponds to an entity of WordNet.

For each dataset, we split the classes into two disjointed parts – *seen classes* and *unseen classes* as in [17]. The former have training samples (images) while the latter have no training samples but are semantically related to the former. Specifically, in ImageNet, 398 animal classes are used as seen classes, each of which contains around 1,000 images, while classes that are one-hop away from the seen ones in WordNet are taken as unseen classes. In AwA, 40 classes are used as seen classes and 10 as unseen classes. AwA unseen classes

Table 4

Statistics of the image sets.

Dataset	Classes #	Attributes #	Seen Classes #	Unseen Classes #	Total Images #
AwA	50	85	40	10	30,475
ImageNet	895	0	398	497	807,307
ImageNet*	473	0	174	299	418,925

are contained in the ImageNet unseen set and several of the seen classes (24 out of 40) overlap with the ImageNet seen set. We extract those animal classes and build a graph with 3969 nodes for experiments. Please see more statistics of the datasets in Table 4.

Specially, in ImageNet, the connection density between seen class nodes and unseen class nodes varies a lot: some seen (unseen) classes whose surroundings contain multiple (e.g., 5) unseen (seen) classes, named as dense connection, while some seen (unseen) classes whose surroundings contain very few (e.g., 1) unseen (seen) classes, named as sparse connection. To evaluate the impact of different connection density on the performance, we extract a subset ImageNet* from ImageNet with all sparsely-connected classes removed, where seen (unseen) classes connect with more than two unseen (seen) classes. The statistics of ImageNet* are also listed in Table 4.

5.1.2. Baselines

The following ZSL methods are used as baselines: **DAP** [6] and **IAP** [6] which utilize the image attributes to model the inter-class relationship, **Devise** [3] and **ConSE** [4] which linearly map the image features learned by CNNs to the word embedding space of class labels, **SYNC** [7] which develops a series of “phantom” classes as bases to connect seen and unseen classes in the word embedding space, **GCNZ** [5] which utilizes GCN and knowledge graph to predict CNN features of each unseen class, and **DGP** [18] which proposes a dense connection scheme of knowledge graph to optimize the knowledge propagation between distant nodes in shallow networks like GCN.

To demonstrate the ability of AZSL in leveraging attention mechanism to optimize the learning of GCN model \mathcal{M}_{GCN} proposed in Section 4.2.2, we implement \mathcal{M}_{GCN} with two recent models proposed in GCNZ [5] and DGP [18], denoting as AZSL-G and AZSL-D respectively.

5.1.3. Model Configuration

We adopt ResNet50 – a successful CNN architecture for image feature learning [62]. We pre-train it using the samples of seen classes in ImageNet. The

Table 5

Performance (%) of AZSL-G, AZSL-D and the baselines on AwA, ImageNet and ImageNet*. [†] indicates the results come from the original paper. “—” means the method can’t be applied to the dataset.

(a). AwA and ImageNet

Model	AwA	ImageNet		
	Hit@1	Hit@1	Hit@2	Hit@5
DAP	41.4 [†]	—	—	—
IAP	42.2 [†]	—	—	—
Devise	54.2 [†]	5.40	8.53	14.02
Conse	45.6 [†]	9.04	13.96	20.53
Sync	54.0 [†]	13.08	20.35	30.80
GCNZ	68.72 ± 0.08	29.31 ± 0.12	47.11 ± 0.13	71.63 ± 0.07
AZSL-G	69.39 ± 0.10	30.57 ± 0.09	48.23 ± 0.10	71.32 ± 0.08
DGP	83.98 ± 0.09	34.47 ± 0.04	51.59 ± 0.07	74.79 ± 0.09
AZSL-D	84.80 ± 0.13	34.81 ± 0.05	51.72 ± 0.07	74.54 ± 0.15

(b). ImageNet*

Model	ImageNet*		
	Hit@1	Hit@2	Hit@5
GCNZ	23.02	43.22	73.95
AZSL-G (us)	25.67	46.84	74.99
DGP	32.67	53.60	79.37
AZSL-D (us)	33.44	54.63	79.89

output of its second to last layer, whose dimension is 2048, are adopted as the classifier. Following GCNZ [5] and DGP [18], AZSL adopts 6 convolutional layers (AZSL-G) and 2 convolutional layers (AZSL-D) respectively, as well as one attention layer with attention weight threshold $\alpha = 0.01$. In training AZSL, the initial learning rate is set to 0.0002, the dropout parameter is set to 0.5, L_2 regularization parameter λ is set to 0.005, and Adam optimizer is adopted. The initial embedding of class nodes in graph uses the word embedding model Glove [63] whose dimension is set to 300. In *EvidenceMining* algorithm, the minimum support and minimum confidence value are specified as 10% and 30% respectively.

In ZSL, **Hits@k**, which is the ratio of samples whose top k scored labels hit the ground-truth label, is widely used for performance measurement. We set k to 1, 2 and 5, where $k = 1$ is widely believed to be the most important [17]. As AwA has only 10 unseen classes, we use **Hits@1** (i.e., accuracy) alone.

5.2. Evaluation of Attentive ZSL Learner

5.2.1. Performance Comparison

As shown in Table 5, the performance of KG-based ZSL methods, including GCNZ [5], DGP [18] and our AZSL, is much higher than that of traditional methods, especially on ImageNet. This verifies that class semantics extracted from a KG is more effective in model-

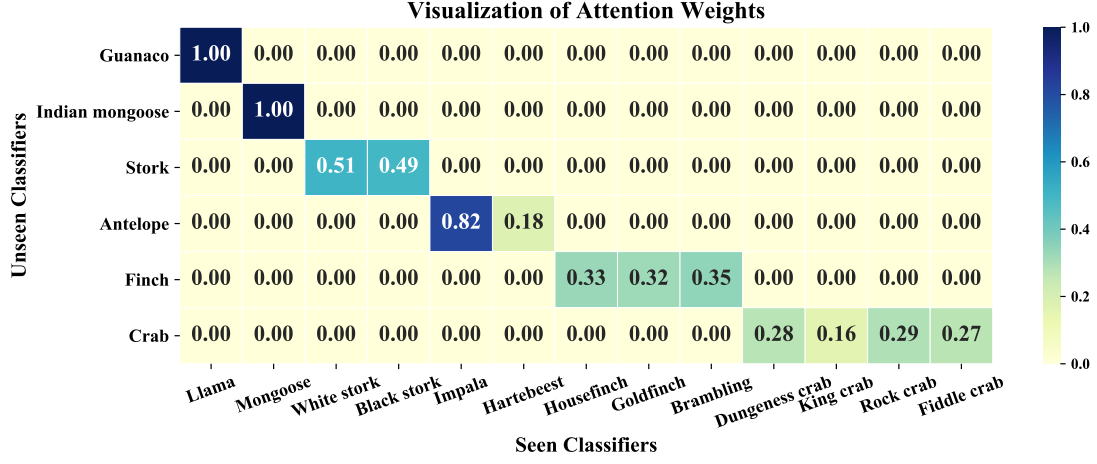


Fig. 6. Impressive seen classifiers (IMSCs) as well as their normalized attention weights of 6 randomly selected unseen classes. 0.00 here means a weight value below a threshold (very close to zero).

ing inter-class relationships and can dramatically improves ZSL performance. This is as expected, because the semantics of class names and attributes utilized by traditional methods is not as rich as that of KG.

Compared with GCNZ and DGP – the state-of-the-art ZSL methods utilizing KG semantics, our AZSL-G and AZSL-D perform better in most of settings. This indicates the effectiveness of our attentive GCN architecture in dealing with the ZSL problem. Considering the main goal of AZSL is to provide human understandable ZSL explanations and it's widely believed that there is a compromise between a machine learning model's interpretation and accuracy [64], the performance improvement of AZSL over GCNZ and DGP is still very promising.

5.2.2. Performance on Dense Graph

We also compare the performance of KG-based ZSL methods on ImageNet*, a dense graph we extract from ImageNet. We find AZSL-G and AZSL-D both have more significant outperformance over GCNZ and DGP respectively. For example, on ImageNet the Hit@1 outperformance rate of AZSL-G is 4.3%, while on ImageNet* it increases to 11.5%. This indicates the superiority of AZSL in dealing with densely connected KG. And it also validates that taking the different contributions of different seen classes into consideration exactly improves the performance of ZSL model.

5.3. Quantitative Analysis of Transferability

In this subsection, we illustrate the feature transferability from seen classes to unseen classes with im-

pressive seen classifiers (IMSCs) learned in Section 4.2.3. Quantitatively, our AZSL also learns fairly good IMSCs.

Figure 6 visualizes some unseen classifiers and their corresponding IMSCs, which displays the relationship between seen and unseen classifiers in transferring deep features in ZSL. The presented examples of IMSCs in Figure 6 are mostly consistent with our common sense on visual features of the animals, for example, in our impression, *Guanaco* and *Llama* are two animals that are similar in appearance. Hence, our AZSL is capable of learning reasonable IMSCs for unseen classifiers and making an elementarily illustration of the feature transferability in ZSL. We also note that the number of IMSCs varies dramatically from one unseen class to another. For example, *Indian mongoose* and *Guanaco* have only one IMSC, while *Finch* and *Crab* have 3 and 4 respectively. According to the statistics, around 79% of unseen classes have only one IMSC, and around 5.6% have two IMSCs.

We further evaluate the impact of IMSCs by analyzing the performance drop when some IMSCs are removed, as shown in Figure 7. Taking results of AZSL-G for example, the performance decreases in all the cases when some IMSCs are removed, in comparison with NO removing. Specially, it drops to 0 in most cases when all IMSCs are removed. This indicates that these IMSCs play a key role in transferring features from seen classes to unseen classes, and they can be the base to generate textual explanations for feature transferability in ZSL.

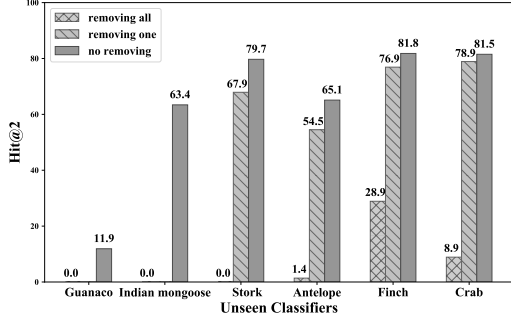


Fig. 7. Hits@2 of AZSL-G when one IMSC is removed, all IMSCs are removed and NO IMSCs are removed.

Regarding the learned IMSCs, they not only elementarily explain the features transferring from seen class to unseen class, so that AI experts can understand, but also can be used for generating textual explanations that common people satisfy.

5.4. Evaluation of Explanations

In this subsection, we demonstrate how human beings are satisfied with the generated explanations. We also compare the impact of different external KGs, and present some case studies.

5.4.1. Human Evaluation

For human evaluation, we invite 25 volunteers without AI expertise to score the generated explanations. Most of them are undergraduate students. We divide unseen class set into 5 parts. Each part has 100 unseen classes and 100 explanations. Each explanation is assessed by 5 volunteers, and the final decision is made by majority voting.

We defined two metrics – *readability* and *rationality* for evaluation. “G” (Good), “M” (Median) or “B” (Bad) are scored for each metric.

Readability: it measures whether an explanation is natural and fluent. Good: fluent, Medium: unnatural but still understandable, Bad: confused and incomprehensible.

Rationality: it measures whether an explanation illustrates the transferability from seen to unseen classes. Good: well illustrated, Medium: insufficient and weakly illustrated, Bad: totally unconvincing.

Note that the two metrics are scored independently. To help volunteers master the domain of the classification application, we prepare some guidelines for each metric, as well as some additional images and external knowledge of classes (e.g., name).

Table 6

Results of human evaluation on the explanations.


Readability		Rationality	
Score	Ratio of Explanations	Score	Ratio of Explanations
G	36.58%	G	73.20%
M	60.77%	M	20.30%
B	2.65%	B	6.50%



Table 6 presents the human evaluation result. We can easily find that explanations for most unseen classes obtain a highly satisfied result, especially on the rationality, and only a very small ratio of explanations get “Bad” on readability and rationality. We find 60.77% of the explanations get Median on readability, but they do not have much impact on people’s satisfaction with rationality. This indicates that the templates for explanation generation need further refinement, which is among our future work.


5.4.2. Impact of Different Types of KGs

We present the generated explanations for unseen classes *Horse*, *Rat*, *Dolphin* and *Stork* in Figure 8, with evidences from the domain-specific Attribute Graph and DBpedia. For example, *Horse* is related with its impressive seen classes *Zebra*, with the attention weight of 1.0. As the upper left case of Figure 8 shows, our explanation generator extracts attribute evidences such as *hooves*, *longneck*, *chewteech*, *tail* from Attribute Graph, to illustrate the common characters between *Horse* and *Zebra* and mine the association rule about them: “{zebra \Rightarrow horse}”, with a support value 73.0% and a confidence value 90.0%. On the other hand, generator also extracts common knowledge from DBpedia: triples such as (*dbr:Horse*, *dct:subject*, *dbr:Equus*) and (*dbr:Zebra*, *dct:subject*, *dbr:Equus*) which describe their common ancestor, and keywords such as *night vision* and *ears* which describe their common characteristics. With these evidences, we can justify that predicting an image as *Horse* is based on the feature transfer from *Zebra*, and is reasonable.

To compare the impact of Attribute Graph and DBpedia, we generate explanations from them separately. Taking *Rat* in Figure 8 as an example, we find that the evidences from Attribute Graph are more likely to generate explanations in vision, including body appearance (*quadrupedal*, *paws*), coat appearance (*fur*), while evidences from DBpedia often generate more general explanations, including not only visual descriptions such as *incisors*, but also common-sense de-

		Horse (Unseen Class)	IMSCs of Horse Zebra (w: 1.0)
Image			
DBpedia Entity		dbr:Horse	dbr:Zebra
Knowledge from Attribute Graph		hooves, longneck, chewteeth, tail, muscle, grazer, plains, grouped, quadrupedal, timid	
		For rule {zebra} \Rightarrow {horse} : sup. = 73.0%; con. = 90.0%	
Knowledge from DBpedia	DBpedia Property/ Relation	(dbr:Horse, dct:subject, dbc:Equus) (dbr:Horse, dct:subject, dbc: Herbivorous_animals)	(dbr:Zebra, dct:subject, dbc:Equus) (dbr:Zebra, dct:subject, dbc:Herbivorous_animals)
	Keywords of Abstract Text	ungulate mammal, domesticated, day and night vision, balance, large ears	black-and-white striped coats, ungulates, social, large ears, night vision
Generated Explanation & Score		The prediction for samples of horse is supported by zebra . <i>They are both grazer, grouped, quadrupedal, both have hooves, muscle, longneck, chewteeth, both live in plains, and behave timid. They are both species of equus, and are both a member of herbivorous animals. They are both ungulate animal, and are similar in night vision and large ears.</i>	
		Readability [G/M/B]: M Rationality [G/M/B]: G	

		Rat (Unseen Class)	IMSCs of Rat		
			Mouse (w: 0.36)	Hamster(w:0.34)	Beaver (w: 0.30)
					
		dbr:Rat	bdr:Mouse	dbr:Hamster	dbr:Beaver
		quadrupedal, ground, nocturnal, paws, small, buckteeth, hibernate, nestspot, agility, furry			
		For rule {mouse, hamster, beaver} \Rightarrow {rat} : sup. = 26.8%; con. = 46.9%			
		(dbr:Rat, hypernym, dbr:Rodents) (dbr:Hamster, hypernym, dbr:Rodents) (dbr:Rat, dct:subject, dbc:Invasive_mammal) (dbr:Mouse, dct:subject, dbc:Invasive_mammal)			
		medium-sized, long-tailed rodents, incisors	pointed incisors, rounded ears, Rodentia, high breeding rate	rodents, sharp incisors, house pets, underground,	semiaquatic rodent, nocturnal, building dams, four incisors
Generated Explanation & Score		The prediction for samples of rat is supported by mouse , hamster and beaver . <i>They are both quadrupedal, nocturnal, small, furry; both have paws, buckteeth, both live in ground, nestspot, both hibernate and behave agility. Rat and hamster are both relevant to rodents via relation hypernym, rat and mouse both belong to invasive mammal. They are both have incisors.</i>			
		Readability [G/M/B]: M Rationality [G/M/B]: G			

		Dolphin (Unseen Class)	IMSCs of Dolphin Killer whale (w: 1.0)
Image			
DBpedia Entity		dbr:Dolphin	dbr:Killer_whale
Knowledge from Attribute Graph		hairless, toughskin, flippers, swims, tail, ocean, grouped, smart, fast, active	
		For rule {killer whale} \Rightarrow {dolphin} : sup. = 48.7%; con. = 62.1%	
Knowledge from DBpedia	DBpedia Property/ Relation	(dbr:Dolphin, dct:subject, dbc:Animals_that_use_echoloc ation)	(dbr:Killer_whale, dct:subject, dbc:Animals_that_use_echoloc ation)
	Keywords of Abstract Text	aquatic, shaped teeth, well- developed hearing, widespread, blubber under skin	oceanic, apex predators, toothed whale, a layer of blubber, excellent hearing, diverse diet
Generated Explanation & Score		The prediction for samples of dolphin is supported by killer whale . <i>They are both hairless, grouped, both have tough skin, flippers, tail, both swim, live in ocean, and behave smart, fast and active. They are both animals that use echolocation. They both have teeth, hearing and blubber.</i>	
		Readability [G/M/B]: M Rationality [G/M/B]: M	


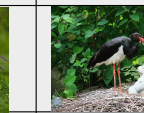
		Stork (Unseen Class)	IMSCs of Stork	
			White stork (w: 0.51)	Black stork (w: 0.49)
				
		dbr:Ciconiiformes	dbr:White_stork	dbr:Black_stork
		white, black, water, wild, fish		
		For rule {white stork, black stork} \Rightarrow {stork} : sup. = 83.3%; con. = 100.0%		
		(dbr:White_stork, dbo:order, dbr:Ciconiiformes) (dbr:Black_stork, dbo:order, dbr:Ciconiiformes)		
		large, long-legged, long-necked, wading, birds, soaring, long, stout bills, migratory	large bird, long red legs, wading, family, Ciconiidae, migrant, carnivore	large bird, wading, family, Ciconiidae, black plumage, long red legs, red beak
Generated Explanation & Score		The prediction for samples of stork is supported by white stork and black stork . <i>They are both white, black, wild, both live in water and eat fish. White stork and black stork both belong to stork biologically. They are both large wading birds, and are similar in long legs.</i>		
		Readability [G/M/B]: G Rationality [G/M/B]: G		

Fig. 8. The explanations of unseen classes: *Horse*, *Rat*, *Dolphin* and *Stork*. In each case, images of the unseen class and its impressive seen classifiers (IMSCs), their matched DBpedia entities, the extracted attributes, triples and keywords are displayed. The association rule of seen and unseen classes for attribute evidences mining are listed with their measurements (“sup.”: support value of the rule, “con.”: the corresponding confidence value). “(w:*)” behind IMSCs denotes the attention weights. The textual explanation as well as the evaluation from volunteers are also displayed. The sentences in italic are explanations generated with attributes, while those marked with underline are generated with triples and keywords.

scriptions such as *rodents* ancestor and *invasive mammal* in biology. The rest of cases follow the same tendency, which may be due to the nature of these two KGs themselves.

We also find and compare some limitations of these two KGs. Due to the great cost on attribute annotations, the scale of Attribute Graph is limited, with only 1,399 objects classes and 588 attributes collected. A considerable portion of classes (about 90%), especially those from ImageNet, whose attributes extracted from Attribute Graph have no more than 10. For example, there is few attribute descriptions about *Stork* and its IMSCs (see the bottom right of Figure 8), the corresponding explanation is not enough to describe the common characters among classes. In contrast, DBpedia owns abundant resources, the knowledge can be accessed as long as the classes in ZSL match the DBpedia entities, which is very friendly for classes in large scale ZSL. However, the keywords extracted from DBpedia’s abstract text sometimes are noisy. It can be found in the case of *Dolphin* (the bottom left of Figure 8). Taking *hearing* as an example, although the extracted keywords of *Dolphin* and *Killer whale* – “well-developed hearing” and “excellent hearing” respectively, describe the same characters about their hearing, only “hearing” is extracted as common keywords because of the different adjective prefix, causing the incompleteness of common knowledge expression. This also motivates us to improve the algorithm for keyword extraction to make full use of the knowledge in abstract text.

In summary, the evidences from Attribute Graph are more applicable to generate explanations for specific domain such as vision, especially when the attribute annotations are sufficient, while the evidences from DBpedia are more general and accessible: it can be applied to different ZSL applications like text classification and large scale ZSL problems with a number of classes. Meanwhile, the explanations from Attribute Graph and DBpedia are compatible with each other, and can be combined.

5.5. Discussion on Feature Transferability and ZSL Prediction

In this section, we analyze the relationship between feature transferability and ZSL prediction according to our explanations. We take the prediction results of AZSL-G as examples to make illustration.

Table 7

Performance (Hit@1 and Hit@2) of AZSL-G according to different types of transferability.

Transferability	Ratio of Unseen Classes	Performance (%)	
		Hit@1	Hit@2
<i>ancestor</i>	49.2 %	25.06	46.79
<i>sibling</i>	38.1 %	29.10	50.58
<i>ancestor-sibling</i>	1.2 %	66.05	79.52
<i>other</i>	11.5 %	37.40	49.56

5.5.1. Successful and Failed Transfer

In ZSL, samples of unseen classes are predicted by transferring features from seen classes. However, we find a case where some unseen classes have no features transferred from seen classes, and the prediction results (Hits@1 and Hits@2) are both 0. We call such a case as a *failed transfer* (FT). In contrast, the case where some unseen classes have effective features transferred from seen classes is named as a *successful transfer* (SF). For example, the Hit@1 and Hit@2 accuracy of unseen class *Eared seal* are both 0 and its feature transferring is failed, while another unseen class *Frog* transferring features from seen class *Tree frog* and *Tailed frog* own performance on Hit@1 with 61.70 and Hit@2 with 79.62, is a successful case.

Those explanations that are scored “Good” on rationality well illustrate the feature transferability from seen classes to unseen classes, and are regarded as *good explanations*, while others are not. We find that 48% of the unseen classes are SF with good explanations, while 39% are SF without good explanations. The shortage of good explanations may be due to (i) the noise of extracted knowledge from the KG, and (ii) incorrect matching between classes and DBpedia entities as mentioned in Section 4.3.2. The former can be solved by developing more advanced methods to extract knowledge, while the later can be improved by traditional ontology alignment systems and modern semantic embedding methods. We also find around 3% of the unseen classes are FT with good explanations. This may be because the learned unseen classifiers are not accurate enough (learning to very low Hit@1 and Hit@2) although features are transferred, or the feature extraction in CNN needs to be refined. This indicates that on the one hand, the class knowledge can be further enhanced, on the other hand, the encoded class knowledge can be utilized to improved the CNN module. The rest (10%) of unseen classes are FT without good explanations. Maybe it’s because some seen classes are missing in the ZSL datasets, resulting in no features can be transferred to these unseen classes.

5.5.2. Different Types of Transferability

We find that the feature transfer from seen classes to unseen classes have different types. For example, some features are transferred between two sibling classes, and some others are transferred from one class to its children. Given a successful transfer between a seen class and an unseen class, we divide it into several types: (i) *ancestor* which refers the case where the seen class is the ancestor of the unseen classes or vice versa (e.g., the unseen class *Stork* is the ancestor of one of the seen classes *White stork* in Figure 8); (ii) *sibling* which refers the case where the seen class and the unseen class are siblings (e.g., the unseen class *Horse* and the seen class *Zebra* in Figure 8 are both children of *Equus*); (iii) *ancestor-sibling*, which refers the case where the relationship between seen and unseen classes includes both ancestor and sibling; (iv) *others* which refers the cases where there are no ancestor or sibling relationship between the seen and unseen classes.

We count all successful transfer predictions of unseen classes in ImageNet according to the different types of transferability. As Table 7 shows, 49.2% of unseen classes learn their classifiers by transferring features from ancestor-neighbor seen classes and own performance on Hit@1 with 25.06, while 38.1% transfer features from sibling-neighbor seen classes and own performance on Hit@1 with 29.10. Nearly 90% of unseen classes focus on these two kinds of feature transfer, it's because the inter-class relationship our model input focuses on the hierarchical structure of classes. We also find that the predictions of classes with *sibling* neighbors are superior to those with *ancestor* neighbors, probably because of the appearance divergence between samples of ancestor classes and descendant classes, while the samples of sibling classes are more similar in appearance. The most notable is that the combination of the two types achieves the highest prediction results. This motivates us that the performance may be improved if we introduce more *sibling*-type feature transfer, such as inputting more sibling-neighbor seen classes, especially together with ancestor neighbors.

6. Conclusion and Outlook

In this study, we investigate explainable ZSL with (1) a new ZSL learner that utilizes inter-class relationships extracted from the KG as well as an Attentive Graph Convolutional Network, and (2) an explanation

generator that can make human understandable explanations with external KGs to justify the feature transferability in ZSL. The study is evaluated with two image sets. We not only achieve higher overall ZSL accuracy than the state-of-the-art baselines, but also generate high quality explanations that can make the feature transfer procedure and ZSL prediction more interpretable. With the explanations, we also investigate the relationship between feature transferability and ZSL performance, which has the potential of further improving the performance of ZSL algorithms.

In the future, we will further improve the explanation quality and ZSL accuracy by utilizing richer semantics from KGs. We may also apply the method in other domains like KG construction and natural language processing.

References

- [1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., Imagenet large scale visual recognition challenge, *International journal of computer vision* **115**(3) (2015), 211–252.
- [2] M. Palatucci, D. Pomerleau, G.E. Hinton and T.M. Mitchell, Zero-shot learning with semantic output codes, in: *Advances in neural information processing systems*, 2009, pp. 1410–1418.
- [3] A. Frome, G.S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov et al., Devise: A deep visual-semantic embedding model, in: *Advances in neural information processing systems*, 2013, pp. 2121–2129.
- [4] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G.S. Corrado and J. Dean, Zero-shot learning by convex combination of semantic embeddings, *arXiv preprint arXiv:1312.5650* (2013).
- [5] X. Wang, Y. Ye and A. Gupta, Zero-shot Recognition via Semantic Embeddings and Knowledge Graphs, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6857–6866.
- [6] 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* **36**(3) (2014), 453–465.
- [7] S. Changpinyo, W.-L. Chao, B. Gong and F. Sha, Synthesized classifiers for zero-shot learning, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5327–5336.
- [8] E. Kodirov, T. Xiang and S. Gong, Semantic autoencoder for zero-shot learning, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3174–3183.
- [9] R.R. Selvaraju, P. Chattopadhyay, M. Elhoseiny, T. Sharma, D. Batra, D. Parikh and S. Lee, Choose Your Neuron: Incorporating Domain Knowledge through Neuron-Importance, in: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 526–541.

- [10] H. Larochelle, D. Erhan and Y. Bengio, Zero-data learning of new tasks., in: *AAAI*, Vol. 1, 2008, p. 3.
- [11] C.H. Lampert, H. Nickisch and S. Harmeling, Learning to detect unseen object classes by between-class attribute transfer, in: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 951–958.
- [12] A. Farhadi, I. Endres, D. Hoiem and D. Forsyth, Describing objects by their attributes, in: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 1778–1785.
- [13] Z. Akata, F. Perronnin, Z. Harchaoui and C. Schmid, Label-embedding for attribute-based classification, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 819–826.
- [14] P. Morgado and N. Vasconcelos, Semantically consistent regularization for zero-shot recognition, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6060–6069.
- [15] 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, pp. 69–77.
- [16] L. Zhang, T. Xiang and S. Gong, Learning a deep embedding model for zero-shot learning, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2021–2030.
- [17] 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* (2018).
- [18] 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*, 2019, pp. 11487–11496.
- [19] P.K. Pushp and M.M. Srivastava, Train Once, Test Anywhere: Zero-Shot Learning for Text Classification, *arXiv preprint arXiv:1712.05972* (2017).
- [20] J. Zhang, P. Lertvittayakumjorn and Y. Guo, Integrating Semantic Knowledge to Tackle Zero-shot Text Classification, *arXiv preprint arXiv:1903.12626* (2019).
- [21] L. Logeswaran, M.-W. Chang, K. Lee, K. Toutanova, J. Devlin and H. Lee, Zero-Shot Entity Linking by Reading Entity Descriptions, *arXiv preprint arXiv:1906.07348* (2019).
- [22] S. Rijhwani, J. Xie, G. Neubig and J. Carbonell, Zero-shot neural transfer for cross-lingual entity linking, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 6924–6931.
- [23] O. Levy, M. Seo, E. Choi and L. Zettlemoyer, Zero-Shot Relation Extraction via Reading Comprehension, in: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, 2017, pp. 333–342.
- [24] P. Pasupat and P. Liang, Zero-shot entity extraction from web pages, in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1, 2014, pp. 391–401.
- [25] O. Biran and C. Cotton, Explanation and justification in machine learning: A survey, in: *IJCAI-17 Workshop on Explainable AI (XAI)*, 2017, p. 8.
- [26] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti and D. Pedreschi, A survey of methods for explaining black box models, *ACM Computing Surveys (CSUR)* **51**(5) (2018), 93.
- [27] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti and D. Pedreschi, A survey of methods for explaining black box models, *ACM computing surveys (CSUR)* **51**(5) (2019), 93.
- [28] Z.C. Lipton, The mythos of model interpretability, *arXiv preprint arXiv:1606.03490* (2016).
- [29] C. Rudin, B. Letham and D. Madigan, Learning theory analysis for association rules and sequential event prediction, *The Journal of Machine Learning Research* **14**(1) (2013), 3441–3492.
- [30] 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*, ACM, 2016, pp. 1135–1144.
- [31] W. Landecker, M.D. Thomure, L.M. Bettencourt, M. Mitchell, G.T. Kenyon and S.P. Brumby, Interpreting individual classifications of hierarchical networks, in: *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*, IEEE, 2013, pp. 32–38.
- [32] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm and N. Elhadad, Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission, in: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2015, pp. 1721–1730.
- [33] O. Biran and K.R. McKeown, Human-Centric Justification of Machine Learning Predictions., in: *IJCAI*, 2017, pp. 1461–1467.
- [34] Q. Li, J. Fu, D. Yu, T. Mei and J. Luo, Tell-and-answer: Towards explainable visual question answering using attributes and captions, *arXiv preprint arXiv:1801.09041* (2018).
- [35] D. Ratcliffe and K. Taylor, Closed-world concept induction for learning in OWL knowledge bases, in: *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2014, pp. 429–440.
- [36] I. Tiddi, M. dAquin and E. Motta, Dedalo: Looking for clusters explanations in a labyrinth of linked data, in: *European Semantic Web Conference*, Springer, 2014, pp. 333–348.
- [37] J. Chen, F. Lécué, J.Z. Pan, I. Horrocks and H. Chen, Knowledge-based Transfer Learning Explanation, in: *KR*, 2018.
- [38] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola and E. Hovy, Hierarchical attention networks for document classification, in: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480–1489.
- [39] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang and G.W. Cottrell, A dual-stage attention-based recurrent neural network for time series prediction, in: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, AAAI Press, 2017, pp. 2627–2633.
- [40] S.J. Pan and Q. Yang, A survey on transfer learning, *IEEE Transactions on knowledge and data engineering* **22**(10) (2009), 1345–1359.
- [41] K. Weiss, T.M. Khoshgoftaar and D. Wang, A survey of transfer learning, *Journal of Big data* **3**(1) (2016), 9.
- [42] N. Zhang, S. Deng, Z. Sun, G. Wang, X. Chen, W. Zhang and H. Chen, Long-tail Relation Extraction via Knowledge Graph Embeddings and Graph Convolution Networks, *arXiv preprint arXiv:1903.01306* (2019).

- [43] C.-W. Lee, W. Fang, C.-K. Yeh and Y.-C. Frank Wang, Multi-label zero-shot learning with structured knowledge graphs, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1576–1585.
- [44] F. Lecue, J. Chen, J.Z. Pan and H. Chen, Augmenting Transfer Learning with Semantic Reasoning, in: *IJCAI*, 2019.
- [45] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, How transferable are features in deep neural networks?, in: *Advances in neural information processing systems*, 2014, pp. 3320–3328.
- [46] T. Liu, Q. Yang and D. Tao, Understanding How Feature Structure Transfers in Transfer Learning., in: *IJCAI*, 2017, pp. 2365–2371.
- [47] Y. Geng, J. Chen, E. Jimenez-Ruiz and H. Chen, Human-centric Transfer Learning Explanation via Knowledge Graph, *arXiv preprint arXiv:1901.08547* (2019).
- [48] G.A. Miller, WordNet: a lexical database for English, *Communications of the ACM* **38**(11) (1995), 39–41.
- [49] 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* **36**(3) (2013), 453–465.
- [50] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie and P. Perona, Caltech-UCSD birds 200 (2010).
- [51] G. Patterson and J. Hays, Sun attribute database: Discovering, annotating, and recognizing scene attributes, in: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 2751–2758.
- [52] O. Russakovsky and L. Fei-Fei, Attribute learning in large-scale datasets, in: *European Conference on Computer Vision*, Springer, 2010, pp. 1–14.
- [53] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer et al., DBpedia—a large-scale, multilingual knowledge base extracted from Wikipedia, *Semantic Web* **6**(2) (2015), 167–195.
- [54] T.N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, *arXiv preprint arXiv:1609.02907* (2016).
- [55] 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*, 2013, pp. 3111–3119.
- [56] R. Agrawal, T. Imieliński and A. Swami, Mining association rules between sets of items in large databases, in: *Acm sigmod record*, Vol. 22, ACM, 1993, pp. 207–216.
- [57] R. Agrawal, R. Srikant et al., Fast algorithms for mining association rules, in: *Proc. 20th int. conf. very large data bases, VLDB*, Vol. 1215, 1994, pp. 487–499.
- [58] P.N. Mendes, M. Jakob, A. García-Silva and C. Bizer, DBpedia spotlight: shedding light on the web of documents, in: *Proceedings of the 7th international conference on semantic systems*, ACM, 2011, pp. 1–8.
- [59] R. Mihalcea and P. Tarau, TextRank: Bringing order into text, in: *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.
- [60] G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross and K.J. Miller, Introduction to WordNet: An on-line lexical database, *International journal of lexicography* **3**(4) (1990), 235–244.
- [61] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, Imagenet: A large-scale hierarchical image database (2009).
- [62] 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*, 2016, pp. 770–778.
- [63] J. Pennington, R. Socher and C. Manning, Glove: Global vectors for word representation, in: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
- [64] H. Paulheim, Make embeddings semantic again!, in: *International Semantic Web Conference (P&D/Industry/BlueSky)*, 2018.
- [65] G. Salton, A. Wong and C.-S. Yang, A vector space model for automatic indexing, *Communications of the ACM* **18**(11) (1975), 613–620.
- [66] H. Liu, Q. Yin and W.Y. Wang, Towards Explainable NLP: A Generative Explanation Framework for Text Classification, *arXiv preprint arXiv:1811.00196* (2018).
- [67] G. Piao and J.G. Breslin, Transfer learning for item recommendations and knowledge graph completion in item related domains via a co-factorization model, in: *European Semantic Web Conference*, Springer, 2018, pp. 496–511.