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

Yuxia Geng [a], Jiaoyan Chen [b], Zhiquan Ye [a], Zonggang Yuan [c], Wei Zhang [d] and Huajun Chen [a,e,*]
[a] *College of Computer Science and Technology, Zhejiang University, 38 Zheda Rd, Hangzhou, China*
E-mail: gengyx@zju.edu.cn, yezq@zju.edu.cn, huajunsir@zju.edu.cn
[b] *Department of Computer Science, University of Oxford, 15 Parks Rd, Oxford OX1 3QD, UK*
E-mail: jiaoyan.chen@cs.ox.ac.uk
[c] *NAIE Product Department, Huawei Technologies Co., Ltd, 101 Software Avenue, Nanjing, China*
E-mail: yuanzonggang@huawei.com
[d] *Bussiness Platform Department, Alibaba Group, 969 Wenyi West Rd, Hangzhou, China*
E-mail: lantu.zw@alibaba-inc.com
[e] *Knowledge Engine Group, AZFT Joint Lab, 1818-2 Wenyi West Rd, Hangzhou, China*

Editor: Dagmar Gromann, University of Vienna, Austria
Solicited reviews: Dagmar Gromann, University of Vienna, Austria; Michael Cochez, Fraunhofer FIT, Sankt Augustin, Germany; Two anonymous reviewers

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 textual explanations that are more acceptable to not only machine learning specialists but also common 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 transferability of features 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 *class knowledge* from WordNet with deep features learned from CNNs (i.e., encode inter-class relationship to predict classifiers), in which the features of unseen classes are transferred from seen classes to predict the samples of unseen classes, with impressive (important) seen classes detected, while the latter generates human understandable explanations for the transferability of features with *class knowledge* that are enriched by external KGs, including a domain-specific Attribute Graph and DBpedia. We evaluate our method on two benchmarks of animal recognition. Augmented by class knowledge from KGs, our framework generates promising explanations for the transferability of features, 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

*Corresponding author. E-mail: huajunsir@zju.edu.cn.

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 classification tasks (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 *a human can recognize new objects through the class knowledge (e.g., description) itself, even without labeled samples*. For example, considering the animal class "*Serval*", even though a human has never seen its 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 infer the 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 word embeddings of class names learned from text corpora for transferring e.g., CNN features, while others (e.g., [5, 6]) prefer more complex knowledge like class hierarchy and class attributes. These methods aim at learning and predicting for unseen classes ([3–8]), but are black-box models: the transferability of features between classes is uninterpretable. This not only limits human's trust in prediction results of ZSL models, considering that ZSL is a method which recognizes the samples of new classes but has never been trained with their labeled samples, but also restricts the potential of improving ZSL models, for example, with explanations, machine learning specialists would master which classes whose features are transferable to learn the features of unseen classes and which are not so that optimizing the ZSL models by adding necessary features or removing inadequate features. Therefore, in this paper, we focus on explaining the transferability of features in ZSL and generating textual explanations which can be understood by not only specialists but also non-specialists.

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 learn the mapping between class attributes and individual neurons in deep networks, and then transfer neurons from seen classes to unseen classes, where class attributes are taken as textual explanations to justify the decisions made by unseen classifiers (cf. more in Section 2.3). Such work indicates that it is feasible to explain ZSL by class knowledge such as class attributes. However, it focuses on grounding the network neurons in interpretable semantics but ignores the feature transferability which is the core of ZSL. Also, its method is ad-hoc, only working for predefined class attributes, while ours supports not only attributes but also general knowledge in different formats, coming from external KGs like DBpedia.

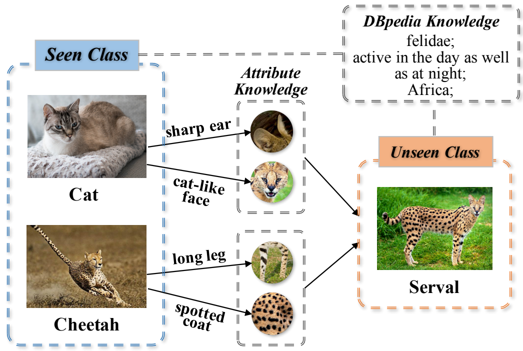


Fig. 1. An example of recognizing *Serval* (unseen class) with two seen classes (*Cat* and *Cheetah*). We focus on explainable ZSL, which extracts domain-specific attributes as well as general knowledge, such as sharp ear, face appearance, long leg, spotted coat and felidae ancestor, as evidence to generate textual explanations that are more understandable by humans.

In this paper, we propose a KG based framework to explain the transferability of features in ZSL. It first adopts a KG named WordNet and an Attentive Graph Convolutional Neural Network (AGCN) to model and encode inter-class relationship for ZSL, which is also known as an *Attentive ZSL Learner* (AZSL). Namely, a matching between inter-class relationship and CNN features is learned. It then uses an *explanation generator* to extract rich class knowledge from a domain-specific Attribute Graph and general external KGs (e.g., DBpedia) as evidence for ZSL explanation. For example, as shown in Figure 1, the attribute knowledge of *sharp ear* and the DBpedia knowledge of *felidae ancestor* are used to illustrate the transferability of features from *Cat* and *Cheetah* to *Serval*. Finally, we propose multiple templates to generate human understandable explanations.

Briefly, our work contributes are as follows:

– 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 transfer 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 explanations with class semantics from not only domain-specific KGs like Attribute Graph but also general KGs like DBpedia.
- A series of templates are designed to organize these class semantics and generate human-consumable natural language explanations.
- Lastly, experiments on two image classification benchmarks are conducted to evaluate the ZSL learner and the explanation generator[1]. Analyses on different metrics and human assessment validate the effectiveness of our method.

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 report the evaluation results. Finally, we conclude the paper and discuss some future directions.

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 are omitted from the training set, with the features from training (seen) classes being transferred. In computer vision, techniques for utilizing 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] annotate each class with a set of attributes and propose two attribute-based classification methods, where the features are transferred between seen and unseen classes via attribute sharing. Recent methods prefer to utilize class embeddings (i.e., the word embeddings of class names) trained on classes' textual descriptions to explore the class semantics [3, 4, 15, 16]. For example, Frome et al. [3] present a *visual-semantic embedding* model, which linearly maps image (visual) features into the class embedding space to predict the labels of images. However, the state-of-the-art performance in zero-shot image classification is achieved by those who utilize KGs for class relationship [5, 17, 18]. For example, Wang et al. [5] use WordNet to model the semantic relationship of hierarchical classes and encode it using GCN to predict classifiers for unseen classes. Considering that graph convolutional operation in GCN only aggregates the features of first-order neighbors, the authors propose to stack multiple graph convolutional layers (e.g., 6) to propagate features towards distant nodes. While Kampffmeyer et al. [18] propose a dense connection scheme that connects distant nodes via additional links to optimize the propagation of features from distant nodes with only 2 convolutional layers. Following the above ideas, we combine class embeddings and class hierarchy as class knowledge to transfer features from seen classes to unseen classes. Unlike the above GCN-based encoder, we propose to utilize Attentive GCN to encode the inter-class relationship, which can assign different importance to different neighboring classes to augment the feature propagation between classes. Moreover, the learned attention weights indicate the most contributing seen classes in the feature transfer, enabling explaining the transferability of features.

There are also some ZSL methods for dealing with the training sample shortage problem in other domains, such as Natural Language Processing (NLP) including text classification [19–21], entity linking [22, 23], relation extraction [24] and machine translation [25]. These methods also introduce high-level knowledge of labels to conduct feature transfer from seen labels to unseen labels. Notably, NLP data and labels are both symbol-based representation, which leads to benefits in feature transfer. In contrast, the feature transfer in our work is more challenging considering the gap between vision and symbol.

2.2. Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI), which aims to produce interpretable models or predictions, is becoming more and more popular nowadays [26–

[1] Code and the Attribute Graph are available at https://github.com/genggengcss/X-ZSL.

28]. Such methods enable humans to understand, trust, and effectively manage AI systems and their decisions. Some of the explainable works design white-box and inherently interpretable models like rule-based systems [29], while others try to justify the prediction of black-box models by for example approximating its behaviour locally with simple interpretable linear models [30], or quantifying the contribution of each single input variable [31].

Some explanations target humans with AI expertise. They can be used for system debug to manage and develop machine learning models efficiently [32, 33]. While some explanations are for common people without AI expertise, for example, they help medical doctors to understand the decisions made by AI-based systems [34]. Most of these works prefer to generate textual explanations, which are more understandable by humans. For example, Biran et al. [35] introduce linguistic expressions from Wikipedia articles to explain the stock price prediction with natural language sentences. Li et al. [36] generate attributes and captions of images to verify whether the system really understands the image content when answering a visual question.

There are also a few works devoted to enriching the explanation with knowledge graphs (KGs), by utilizing human understandable background knowledge and common sense in these KGs, as well as their underlying semantics that can be inferred by reasoning [37, 38]. For example, Tiddi et al. [38] exploit Linked Data as background knowledge to generate explanations for data clusters. Chen et al. [39] extract evidence from local domain ontologies and external KGs like DBpedia using Semantic Web techniques to explain the results of flight delay forecasting.

Another related direction is to utilize the attention mechanism to explain [40, 41]. For example, Yang et al. [40] leverage attention layers to select words and sentences that have a decisive effect on document classification as explanations. Our work also utilizes such an attention technique but goes beyond it. It includes a general framework to incorporate class semantics from KGs and generate textual explanations for the core of ZSL – the transferability of deep features.

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 [42, 43]. ZSL algorithms usually transfer features learned by deep neural networks from seen classes (domains with labeled training samples) to predict the 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 [39, 44–46]. For example, in [46], prior knowledge about the prediction tasks and domains are expressed by ontologies and further utilized to analyze the transferability of features and samples to augment transfer learning. For another example, Zhang et al. [44] propose a transfer learning based algorithm for long-tail relation extraction, which incorporates data features from data-rich relations to tackle the prediction of data-poor relations. Knowledge of the relation, which comes from a KG, is investigated to enhance the feature learning of 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 study, we not only utilize KGs for performance improvement (i.e., 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 [39, 47–49]. For example, Liu et al. [48] assume that the features are transferable from a source domain to a target domain if the source and target domains have similar feature structures. Chen et al. [39] extract knowledge (e.g., ontology axioms and DBpedia facts) that co-exist in the source and target domain to explain the transferability of features learned by deep neural networks. These works indicate that the transferability of features is highly related to the knowledge of the source and target domain. In this paper, we also focus on extracting domain knowledge (i.e., class knowledge) to generate explanations for feature transferability. Different from the above works, we on the one hand develop a general framework that generates explanations from different knowledge resources from multiple KGs such as Attribute Graph and DBpedia. On the other hand, we focus on KG-based ZSL – an essential and popular transfer learning branch whose current solutions are all black-box models without explanations.

Few works have been found to explain ZSL with human understandable knowledge. The only work we know is by Selvaraju et al. [9]. It first learns a mapping between class attributes and individual neurons in a network, and then predicts unseen neurons based on the attributes of unseen classes to optimize the learning of unseen classifiers. It also generates textual explanations by inversely mapping the predicted neu-

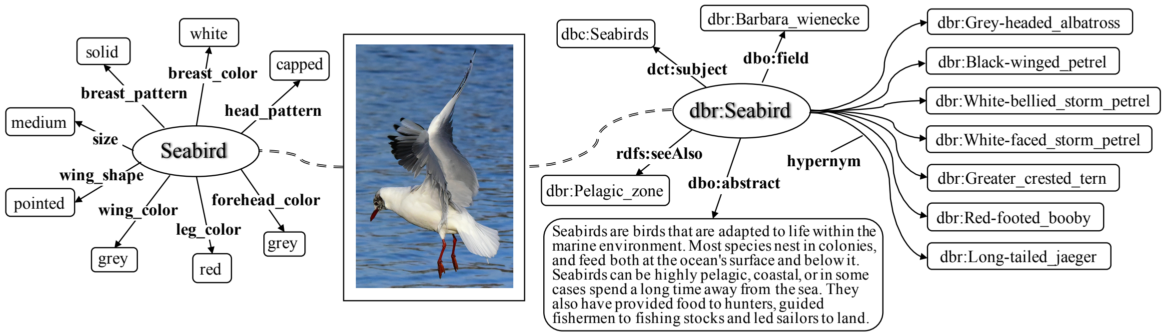


Fig. 2. An example of our two introduced KG resources for animal class *Seabird*. [Left] is the domain-specific Attribute Graph with corresponding entity *Seabird*; [Right] is the general DBpedia with aligned entity *dbr:Seabird*.

rons to class attributes to validate the decisions made by unseen classifiers. The authors focus on grounding the transferred neurons in interpretable semantics, however, ignore the transferability of features between classes in ZSL. By contrast, the explainable ZSL proposed in our work pays attention to the transferability of deep features, which is more important for analyzing the nature of ZSL. Also, in [9], the generation of explanations relies on the input class attributes, while our method can access external resources and generate explanations involving not only domain-specific attributes but also general knowledge, which are more expressive in comparison with the ad-hoc attributes. Besides, the input class attributes also have an impact on the prediction of the ZSL model, that is to say, any changes for improving the diversity or quality of explanations (i.e., improving the input attributes) may hurt the classification performance. In contrast, the generation of explanations in our method is relatively independent of the classification model, which is more flexible than other explainable methods that need to make a tradeoff between accuracy and interpretability.

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 denoted 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. For each class, a unique classifier will be trained to predict whether a sample is of the class or not. ZSL aims to learn classifiers for unseen classes by transferring features learned from $\mathcal{D}_{tr}$ based on the semantic relationship between seen and unseen classes.

There are usually two prediction settings in ZSL: the standard ZSL and the generalized ZSL [17]. The former is to predict the labels of testing samples in $\mathcal{D}_{te}$ with candidate labels from U, while the latter is a more challenging but a more realistic case, which predicts the testing samples of seen and unseen classes with candidate labels from both seen and unseen classes (i.e., $S \cup U$). In our experiments, we evaluate the ZSL model under both two settings to validate the effectiveness of our AZSL.

3.2. Class Knowledge

In our study, we introduce three kinds of Knowledge Graphs (KGs) to describe the *class knowledge*, which depicts the semantic relationship between classes. These KGs are used for transferring features in ZSL model as well as generating explanations for the feature transferability. We briefly introduce them below.

WordNet [50] is a lexical knowledge base of 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 the attribute annotations of classes. These annotations describe the characteristics of classes, especially the visual ones, such as the color, shape and important parts of objects. For each class, we organize its attribute annotations in the form of triples (c, a, v), where c represents the class, a represents the attribute item of this class and v is the corresponding item value. Taking class *Seabird* as an example, as Figure 2 [Left] shows, its attribute annotation "grey wing color" can be described as (*Seabird, wing_color, grey*). We collect these annotations from existing ZSL datasets (e.g., AwA [6] and CUB [51]) and Wikipedia descriptions of classes. The constructed KG contains 1,399 animal classes, 588 attributes and 8,114 triples in total. It is now available in our GitHub repository. Notably, the scale of Attribute Graph is small compared with other public KGs, we look forward to augmenting it by crowdsourcing in the future. Besides, such attribute annotations are available in animal recognition tasks, while in other tasks or domains, they can come from domain knowledge or experts.

DBpedia [52] is a general knowledge graph which includes common sense and background descriptions of classes. With knowledge from Wikipedia encyclopedia, DBpedia is a large scale KG consisting of 4.58 million entities and 3 billion facts. Animal classes in ZSL can be matched to entities in DBpedia. For example, as shown in Figure 2 [Right], class *Seabird* is aligned with entity *dbr:Seabird*. Different from Attribute Graph, DBpedia contains more general knowledge, including the textual description of entity (i.e., abstract text) from property *dbo:abstract*, the semantics between entities linked by different relations (e.g., *hypernym*), etc. In our experiments, we use public DBpedia SPARQL Endpoint query service, which loads 2016-10 DBpedia dump, to access the DBpedia resources (more details at https://wiki.dbpedia.org/public-sparql-endpoint).

In order to provide an overall explanation for ZSL in animal recognition task, we take Attribute Graph as well as DBpedia as external KGs to extract visual knowledge like "red leg color" as well as general knowledge like descendants of "Seabird" in biology.

4. Methodology

4.1. Framework Overview

In this paper, we present a KG-based framework to explain the transferability of features in ZSL in a human understandable manner, including an Attentive ZSL learner (AZSL) and an explanation generator, as shown in Figure 3. AZSL first models the hierarchical relationship of seen classes, unseen classes as well as their ancestor and descendant classes using WordNet, and then matches this class knowledge with deep features extracted from CNNs, which pursue class discrimination and are the core components of a classifier. Specifically, we utilize AGCN to encode the inter-class relationship and then predict a CNN classifier for each class, in which the unseen classifiers are learned by transferring features from seen classifiers. Considering that different seen classes have different contributions in the feature transfer, we introduce an additional attention layer in AGCN to learn the attention weights of seen classes. In this way, we select the most contributing seen classes for unseen classes as well as master the transfer of features from seen classes to unseen classes. Next, given unseen classes and their contributing seen classes, the explanation generator extracts richer class knowledge from external KGs, such as class attributes, semantic relations between classes and textual descriptions of classes, as evidence to justify why these seen classes transfer their features to the unseen ones (i.e., the transferability of deep features from seen classes to unseen classes). The generator also generates natural language explanations with these evidence using some hand-crafted templates.

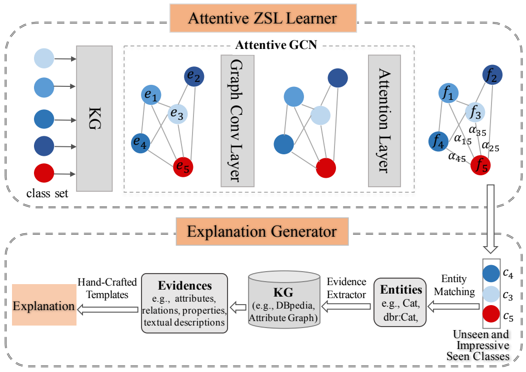


Fig. 3. Our proposed KG-based explainable ZSL framework.

4.2. Attentive ZSL Learner

AZSL utilizes the class knowledge from WordNet and an Attentive Graph Convolutional Network (AGCN) to predict classifiers for unseen classes. As Figure 4 shows, It first pre-trains a discriminative *CNN classifier* for each seen class (Figure 4 [Right]), and then encodes class knowledge to predict classifiers for classes especially for unseen classes (Figure 4 [Left]).

4.2.1. CNN Classifier

Consider Convolutional Neural Network (CNN), a frequently used network for feature extraction in object

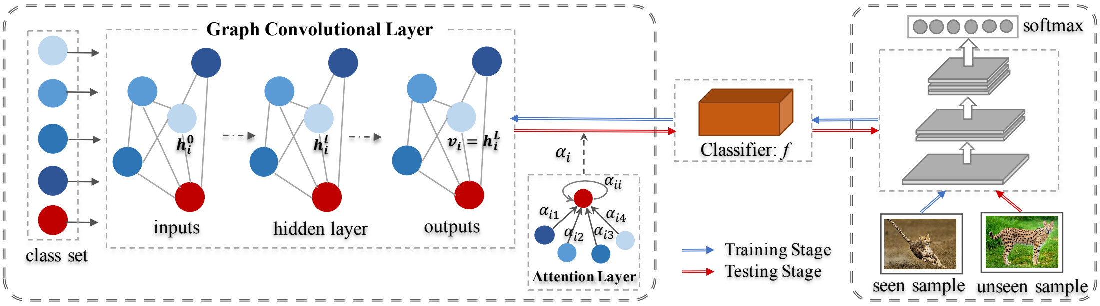


Fig. 4. Overview of attentive ZSL learner. During training, AZSL encodes class knowledge to predict classifiers with multiple graph convolutional layers and an attention layer; During testing, predicted classifiers are used to conduct nearest neighbor search to classify the testing images.

recognition, where significant features of images are extracted to make predictions. Given an object class, when we use its images to train a CNN, the second to the last layer of the network will output a set of class-specific parameters. These parameters constitute a real-valued vector representing the discriminative visual features of this class, which can be used to classify new images of this class. Therefore, in this paper, we take this vector as a classifier and use it to classify the testing images by performing nearest neighbor search (more prediction details are in Section 4.2.3).

As a result, in AZSL, we hope to leverage AGCN to predict such a classifier for each class especially for each unseen class. In particular, as seen classes have enough training samples to learn their classifiers, we pre-train a set of seen classifiers with samples from $\mathcal{D}_{tr}$ as ground truth to supervise the training of AGCN. After training, the classifiers of unseen classes can be learned to predict the samples from $\mathcal{D}_{te}$.

4.2.2. Predicting Classifiers

AGCN is used to encode the graph-structured inter-class relationship and predict a classifier for each class node. It includes multiple graph convolutional layers and an attention layer.

Graph Convolutional Layer is to conduct the convolutional operation on a graph, which propagates information between nodes and captures the dependency of graph-structured data. In each convolutional layer, the convolutional operator computes a node's hidden feature by aggregating features from its neighboring nodes defined in the graph, and updates it to the next layer. Mathematically, given a class node i, its hidden feature at l-th layer is learned as follows:

$$h_i^l = \sigma(W_l \sum_{j \in \mathcal{N}_i} \frac{h_j^{l-1}}{|\mathcal{N}_i|} + B_l h_i^{l-1}) \tag{1}$$

where $\mathcal{N}_i$ is the set of neighboring classes of class i. According to the optimizer in graph convolutional operation, the neighboring classes here mean the first-order neighbors of i. W_l and B_l represent the layer-specific weight matrix and bias term, respectively. $\sigma(\cdot)$ denotes the activation function such as LeakyReLU (More details are in [53]).

Stacking the convolutional layer one after another, we can output the feature of class i at last layer: $v_i = h_i^L$, with the features of other classes encoded. Motivated by [3, 4], we use pre-trained word embeddings of class names to initialize the class nodes. These *embeddings*, learned from text corpora [54], are semantically meaningful representations of classes. Meanwhile, we implement the convolutional operation with models proposed in [5] and [18], in which the different adjacency matrices of graph that indicate the class neighbors are defined and different numbers of convolutional layers are used. We also compare the performance of different graph convolutional operations.

Attention Layer. In the aggregation of graph convolutional layer, we find that different neighboring classes have different impacts on the feature learning of a class. Therefore, in this paper, we propose to utilize attention mechanism – stacking an attention layer after graph convolutional layers to assign attention weights to the neighboring classes and analyze the different contributions of different classes.

Specifically, for each neighboring class of class i, its attention weight is computed by the similarity between its feature vector and v_i, because when a neighboring class contributes more to class i in the aggregation, their features are more similar. For a neighboring class j, its attention weight is computed as:

$$\alpha_{ij} = \frac{exp(cos(v_i, v_j))}{\sum_{k \in \mathcal{N}_i} exp(cos(v_i, v_k))} \tag{2}$$

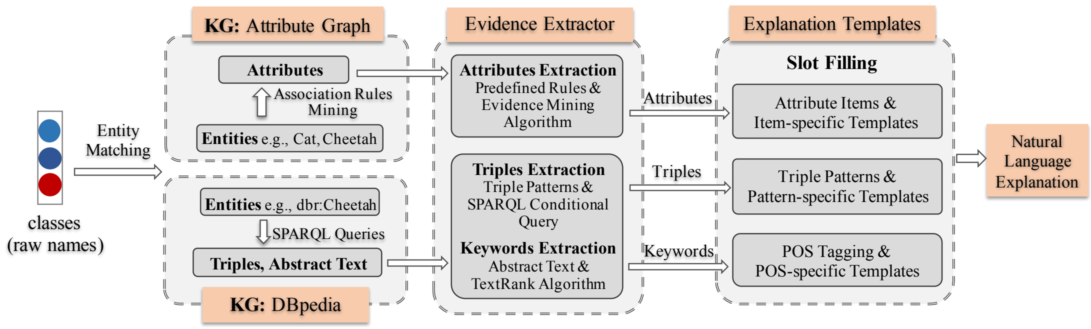


Fig. 5. Illustrating for generating natural language explanation from external Attribute Graph and DBpedia.

where $cos(\cdot)$ denotes the cosine similarity, $\mathcal{N}_i$ is the set of neighboring classes of class i, including class i itself. The computed attention weights are used to update the feature vector of class i as:

$$\bar{v}_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij} \cdot v_j \tag{3}$$

where $\mathcal{N}_i$ also denotes the set of neighboring classes of class i, including class i itself.

It is noted that we stack the attention layer after multiple graph convolutional layers, as Figure 4 shows. We have attempted to add an attention layer after each convolutional layer in the preliminary experiments, however, we found the model is hard to converge. It may be because *(i)* the dimension of hidden features in our model (e.g., $2,048$) is large compared with that in other graph attention networks (e.g., 8 in [55]), or *(ii)* our model is a regression model, whose training is often more difficult than other graph attention networks that are classification models (e.g., [55]).

Training. As previously illustrated, the feature vector of each class node we want the AGCN to output is a classifier that represents class-specific visual features. Therefore, we use pre-trained seen classifiers (cf. Section 4.2.1) to supervise the learning of feature vectors of classes. Specifically, for $|S|$ seen class nodes ($|S|$ here means the number of all seen classes), we have predicted classifiers $\bar{v}_{1 \ldots |S|}$ and pre-trained classifiers $f_{1 \ldots |S|}$, the mean square error between them is computed as the loss function to train the model:

$$\frac{1}{|S|} \sum_{i=1}^{|S|} \mathcal{L}_{MSE}(f_i, \bar{v}_i) \tag{4}$$

Obviously, the model is trained in a semi-supervised manner. For unseen classes, their classifiers can be inferred (learned) by aggregating visual features from their neighboring seen classes.

4.2.3. *Predicting Testing Samples*

With predicted classifiers, we perform nearest neighbor search to predict labels for testing samples. Specifically, at test time, when a testing image arrives, AZSL first extracts its features using pre-trained CNN, and then multiplies the image features with these classifiers to produce some similarity scores. The class corresponding to the most similar classifier (i.e., the nearest one) is the predicted label. Regarding different prediction settings in ZSL, the candidate classifiers involve unseen classifiers (i.e., $\{\bar{v}_i\}_{i=1}^{|U|}$) and the testing images are from unseen classes when it is standard ZSL; while in generalized ZSL, the candidates involve both seen and unseen classifiers (i.e., $\{\bar{v}_i\}_{i=1}^{|S|+|U|}$) and the testing images are from both seen and unseen classes.

Meanwhile, we can learn contributing seen classes for each unseen class from attention layer. These seen classes have high attention weights and each of them is believed to be important in transferring features to the unseen class. We name them as **impressive seen classes** (IMSCs in short). In this way, we automatically detect seen classes that have decisive effects on the feature learning of unseen classes, which is the basis for analyzing the transferability of features in ZSL.

4.3. *Explanation Generator*

Given unseen classes and their impressive seen classes, we introduce two external knowledge graphs, the domain-specific Attribute Graph and the general DBpedia, to extract reliable evidence and generate human understandable explanations to justify the feature transferability between seen and unseen classes.

The explanation generation procedure is illustrated in Figure 5. Briefly, we *(i)* match raw class names with entities of external KGs; *(ii)* adopt different strategies to extract supported evidence from different external KGs; and *(iii)* generate natural language explanations with some templates.

4.3.1. Domain-specific KG: Attribute Graph

Considering that ZSL classes are naturally matched with entities in Attribute Graph, we first begin by extracting evidence. In Attribute Graph, the evidence refers to the common attributes shared by unseen classes and their impressive seen classes. However, the searching space is often large for finding the 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 the association rules of classes, with an algorithm named *EvidenceMining* proposed. The mined rules illustrate the semantic association between seen and unseen classes, and the supporting set of a rule is a set of common attributes shared by these classes, which are desired evidence to explain the feature transferability from seen classes to unseen classes.

Association rule mining is widely used in Data Mining. It was first proposed for mining the association rules of shopping 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 a set of classes that both have a specific attribute. The rule of classes means that these classes are associated because they share a set of identical attributes.

Let $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ and $\mathcal{C} = \{c_1, c_2, ..., c_m\}$ 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 an attribute a_i and consists of a set of classes C that both have 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 has *support value* $s\%$ when $s\%$ of attributes in $\mathcal{D}$ are shared by the classes in $X \cup Y$, denoted as $support(X \cup Y)$, accordingly, the support set of the rule refers to a set of common attributes shared by the classes in $X \cup Y$. Also, the *confidence value* of the rule $\{X \Rightarrow Y\}$ is computed as: $c\% = support(X \cup Y)/support(X)$, meaning that among all attributes shared by the classes in X, $c\%$ of them are also shared by the classes in Y. For an unseen class u and its impressive seen class set $S = \{s_1, ..., s_n\}$, the potential association rule of them can be predefined as:

$$\begin{aligned} \{s_1\} &\Rightarrow \{u\} \\ &... \\ \{s_n\} &\Rightarrow \{u\} \\ \{s_1, ..., s_n\} &\Rightarrow \{u\} \end{aligned} \tag{5}$$

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 Class sets

Class set	Support Set (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%

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 frequent class sets (i.e., the sets of classes whose support values are large than the specified minimum support value) and their support sets are firstly mined as Table 1(b) shows, from which the rules of these classes can be generated. Specifically, for each frequent class set (containing more than one class), the classes in which are randomly split into two parts, one is included in X while the other is included in Y or vice versa, and form a temporary rule $\{X \Rightarrow Y\}$. After traversing all possible splits, a set of temporary rules can be generated. Then, the confidence values of these temporary rules will be computed, those whose confidence values are large than the specified minimum confidence value will be output as the mined rules. The mining of frequent class sets and rules can be implemented by existing asso-

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 evidence for u and S;
1: $\mathcal{C} = \{u, s_1, ..., s_n\}$; % *Prepare class set*
2: $\mathcal{A} = \emptyset$; % *Init. of attribute set*
3: **for** each class $c \in \mathcal{C}$ **do**
4: % *Get attributes of c*
5: $\mathcal{A}_c$ = ExtractAttribute($\mathcal{G}$, c);
6: Append $\mathcal{A}_c$ to $\mathcal{A}$;
7: **end for**
8: $\mathcal{D}$ = ConstructDataset($\mathcal{G}, \mathcal{C}, \mathcal{A}$); % *Store Trans.*
9: $k = n + 1$; % *Size of frequent class set*
10: % *Mine frequent class sets F, return their support values V_s and support sets F_s; Mine rule set R_C and return their confidence values V_c*
11: F, V_s, F_s, R_C, V_c = **Apriori**($\mathcal{D}, k, s_{min}, c_{min}$);
12: R_u = Instantiate($R, \mathcal{C}$); % *Instantiate predefined rule set for u*
13: % *Filter mined rule set and Output desired rules*
14: R'_u = Filter(R_C, R_u);
15: % *Output the support sets of rules as evidence*
16: A = Get(R'_u, F_s);
17: **return** A;

ciation rule mining algorithms such as Apriori [57]. As a result, as Table 1 shows, we mine a rule like "$\{$*Polar bear*, *Raccoon*$\} \Rightarrow \{$*Grizzly bear*$\}$", as well as its support set: $\{$*claws, furry*$\}$ which contains the common attributes shared by these three classes and can be taken as evidence to justify the transferability of features from *Polar bear* and *Raccoon* to *Grizzly bear*.

Algorithm 1 illustrates the pseudocode of mining rules and common attributes. Given an unseen class and its impressive seen classes, we first extract attributes of each class from Attribute Graph $\mathcal{G}$ to construct the transaction database. Then, the Apriori algorithm is applied to mine frequent class sets and rules, with support value and confidence value constraints. The mined rules cover all possible rules of classes we input, however, only those from seen classes to unseen classes are what we need. Therefore, we utilize the rule set predefined in Eq. (5) to filter these mined rules and output those from impressive seen classes to unseen classes. Besides, for each frequent class set, Apriori can simultaneously generate its support set (i.e., common attributes across the classes in the frequent class set), therefore, for each filtered rule, we also output its support set, which contains the common attributes shared by the unseen class and its impressive seen classes, and is the evidence we desire.

In this way, we not only mine the association rules of seen and unseen classes with some measurements produced, e.g., *support value* and *confidence value*, but also extract common attributes from Attribute Graph as evidence to explain the transferability of features from seen classes to unseen classes.

4.3.2. General KG: DBpedia

Match ZSL class with DBpedia Entity. Different from Attribute Graph whose entities can be directly aligned with ZSL classes by name, the matching between DBpedia entities and ZSL classes is 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 DBpedia Lookup service[2], which is based on the index of DBpedia Spotlight [58]. Specifically, we take 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*"[3]. In our preliminary experiments, we have tried to use embedding based methods such as word2vec [54] to compare the word vectors of ZSL classes and DBpedia entities, however, the performance of these methods is unsatisfactory in terms of efficiency and accuracy. Instead, DBpedia Spotlight has a good ability to link unstructured resources to DBpedia data, based on which the online lookup service can immediately and accurately return the DBpedia entity when entering a ZSL class name string.

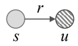
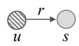
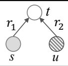
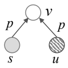
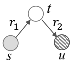
One challenge of class to entity matching is that only a part of ZSL classes have entity correspondences. On the one hand, the entity corresponding to a class may not exist in the KG. For example, DBpedia only contains an entity for *Chicken* but no *Cock* and *Hen*. The latter two however have totally different appearances. On the other hand, some classes are wrongly matched with DBpedia entities. For example, *Red fox* is incorrectly matched with entity *dbr:Fox*, while the correct matching should be *dbr:Red_fox*. To ensure the correctness of class to entity matching, we manually check the matching results and remove the incorrect ones. There are also some algorithms developed to automatically evaluate the entity matching results, for example, the string similarity based methods

[2]https://github.com/DBpedia/lookup

[3]dbr, dbo, etc. are URI prefixes in DBpedia. Please see http://DBpedia.org/sparql?help=nsdecl

Table 2

Triple patterns and corresponding SPARQL query items, where s, u represent entity patterns corresponding to seen and unseen classes 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 to u via relation r.
(u, r, s)		SELECT $?r$ WHERE $\{u\ ?r\ s.\}$	u is directly related to s via relation r.
$(s, r_1, t) \wedge (u, r_2, t)$		SELECT $?r_1\ ?r_2\ ?t$ WHERE $\{s\ ?r_1\ ?t.\ u\ ?r_2\ ?t.\}$	s and u both related to entity t via relation r_1 and r_2, respectively (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 $?r_1\ ?r_2\ ?t$ WHERE $\{\ s\ ?r_1\ ?t.\ ?t\ ?r_2\ u.\}$	s and u is related via a transitional entity t.

[59, 60] and the embedding based methods [61, 62] in entity alignment tasks. However, considering that these methods have some limitations and errors in real-world applications (i.e., relying on well-designed matching patterns or labeled entity pairs), and our work focuses on studying the interpretability in ZSL, we decide to manually check all matching results. It is expected that some methods can be developed to release the pressure of manual verification in the future.

Different from the attribute annotations in Attribute Graph, the knowledge in DBpedia is massive and diverse. Therefore, with matched entities, we utilize SPARQL queries[4] to retrieve two kinds of evidence: *(i)* **abstract text** which is an overall description of entity with keywords included, and *(ii)* structured **triples** which describe the fine-grained semantics of an entity, e.g., properties and relations with other entities.

Extract Triples. 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). From these triples, we can find some correlations between seen and unseen entities (classes), which can be taken as evidence to illustrate the transferability of features from seen to unseen classes. However, only a portion of triples are useful for describing the correlations, we need a method to extract them efficiently.

To this end, we design some *triple patterns*, as shown in Table 2. Based on these patterns, SPARQL queries are developed to retrieve triples, from which the relations or entities that associate seen and unseen entities (classes), and the common properties shared by seen and unseen entities (classes) are extracted. Consider the example in Figure 1, where *Cat* and *Serval* share the same ancestors *Felidae*. The fact can be verified by triples *(dbr:Cat, hypernym, dbr:Felidae)* and *(dbr:Serval, hypernym, dbr:Felidae)*, which are extracted according to the pattern $((s, r_1, t) \wedge (u, r_2, t))$. Notably, some extracted triples or entities are not detailed enough to describe the correlation between entities. For example, the entity *dbc:Birds_of_Europe* in the triple *(dbr:Ruff, dbo:family, dbc:Birds_of_Europe)* is a rather broad concept. This may bring useless information and hurt the quality of generated explanations.

Extract Keywords from Abstract Text. The abstract text of an entity can be directly accessed by SPARQL queries since it is the data value of property *dbo:abstract*. It describes the representative characteristics of entities, especially visual characteristics. However, some descriptions in abstract text are less informative: e.g., the 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* but does not mention much useful information.

Therefore, we adopt TextRank [63], an unsupervised automatic summarization algorithm, to extract keywords from abstract text. The extracted words and phrases are core and descriptive to represent the class-specific properties so that illustrating the knowledge shared between entities. For example, the extracted keyword *Africa* in Figure 1 illustrates the same living environment of *Cheetah* and *Serval*.

4.3.3. Template-based Explanation Generator

Aforementioned extracted items, including attributes from Attribute Graph, as well as triples and keywords

[4] https://www.w3.org/TR/rdf-sparql-query/

Table 3

Templates for generating natural language explanations. An overall illustration is first provided. $\boldsymbol{s, u, t, r, p, a}$, **ADJ**, etc. are slots in templates to be filled. The Left of table is for attributes, where part of attribute items are listed. The Center is for structured triples. The Right is for textual keywords, where all Parts-of-Speech (POSs) of keywords (e.g., adjective (ADJ) and noun (NOUN)) and types of named entities (e.g., LOC) used are listed. Notably, the POS tags attached to the attribute items (e.g., *coat (ADJ)*) mean the attribute values with different POSs.

Overall Illustration: The prediction for samples of $\boldsymbol{u}$ is supported by $\boldsymbol{s}$.

Attributes & Templates		Triples & Templates		Keywords & Templates	
Attribute Item	**Template**	**Triple Pattern**	**Template**	**POS of Keyword**	**Template**
color, size, species, coat (ADJ), ...	They are both $\boldsymbol{a}$.	(s, r, u), (u, r, s)	$\boldsymbol{s(u)}$ is $\boldsymbol{r}$ of $\boldsymbol{u(s)}$.	ADJ	They are both **[ADJ]** animal/bird/other.
body part, shape, coat (NOUN), ...	They both have $\boldsymbol{a}$.	$(s, r_1, t) \wedge (u, r_2, t)$, $(s, r_1, t) \wedge (t, r_2, u)$	$\boldsymbol{s}$ and $\boldsymbol{u}$ are both relevant to $\boldsymbol{t}$ via relation $\boldsymbol{r_1}, \boldsymbol{r_2}$. (or $\boldsymbol{s}$ and $\boldsymbol{u}$ are both a member of $\boldsymbol{t}$.)	NOUN	They both have **[NOUN]**. (or they are similar in **[NOUN]**.)
feeding, habitat	They both eat (or live in) $\boldsymbol{a}$.	$(s, p, v) \wedge (u, p, v)$	$\boldsymbol{s}$ and $\boldsymbol{u}$ share the same $\boldsymbol{v}$ of property $\boldsymbol{p}$.	ADJ+NOUN	$\boldsymbol{s}$ and $\boldsymbol{u}$ are similar in **[ADJ+NOUN]**. (or $\boldsymbol{s}$ and $\boldsymbol{u}$ both have **[ADJ+NOUN]**.)
behaviour, habits	They both behave (or like) $\boldsymbol{a}$.	$(s_1, r_1, u) \wedge (s_2, r_2, u)$	$\boldsymbol{s_1}, \boldsymbol{s_2}$ both belong to $\boldsymbol{u}$. (or $\boldsymbol{s_1}, \boldsymbol{s_2}$ are both species of $\boldsymbol{u}$.)	named entity (LOC, GPE)	They both live in **[LOC].**

from DBpedia, constitute fine-grained class knowledge which can be taken as evidence to explain the transferability of features from seen classes to unseen classes. To make these evidence more understandable, we organize them with some hand-crafted templates.

Inspired by *Slot Filling*, a popular method of completing sentence in dialogue system, we design templates with classes, entities, attributes, relations, properties and keywords as slots, and take extracted items as values to fill in. We design three different kinds of templates, as shown in Table 3, for structured triples, unstructured attributes and keywords respectively.

Attributes are domain-specific descriptions used for annotating objects. The attribute values of the same attribute item describe the same aspects of objects. For example, attributes like *head*, *tail*, *claws* and *leg* describe the body parts of animals, which both belong to the attribute item *body part*, while attributes like *red*, *green* and *blue* describe the appearance colors of animals, which belong to the attribute item *color*. Considering that the attributes belonging to the same attribute item can be expressed in a similar way, we design templates based on attribute items. For example, the common attribute *sharp ear* of *Cat* and *Serval*, which belongs to the attribute item *body part*, can be expressed with the sentence: "They both have sharp ears". Table 3 lists some attribute items and their corresponding templates. Notably, the number of extracted common attributes varies a lot. In order to restrict the length of generated sentences and avoid excessively repetitive expressions, we randomly select 10 attributes to represent the knowledge between seen and unseen classes when the number of common attributes exceeds 10.

Structured triples have fixed formats, especially those extracted using the same *triple patterns*. Thus, we design templates to textualize triples according to the *triple patterns* as shown in the center of Table 3. The entities, properties and relations in extracted triples are taken as values to fill the corresponding slots in templates. For triples *(dbr:Cat, hypernym, dbr:Felinae)* and *(dbr:Serval, hypernym, dbr:Felinae)* extracted by *pattern* "$(s, r_1, t) \wedge (u, r_2, t)$", the following sentence can be generated: "*Cat* and *Serval* are both relevant to *Felinae* via relation *hypernym*". Note that the DBpedia prefixes such as "dbr" in the triple will be removed when generating sentences.

Keywords extracted from abstract text are natural expressions consisting of adjectives, nouns, their combinations and so on. One example is *spotted coat*. Therefore, we design templates based on the parts-of-speech (POSs) of keywords and utilize *POS Tagging* to generate natural language sentences. Specifically, each keyword is first labeled with a POS tag, and then filled into the template with the same POS slot. For the keyword *spotted coat* labeled with an ADJ-NOUN tag, we can use the template with ADJ-NOUN slot to generate the sentence: "They are similar in spotted coat". Additionally, some nouns have special meanings, for example, *Africa* describing the habitat of *Serval* is a location noun. To express them better, we further use Named Entity Recognition (NER) [64] to identify the named entities among these nouns and classify them into different types, and then design different templates regarding different types. The right of Table 3 lists the types we adopt and their corresponding templates. In our experiments, we utilize SpaCy[5], a frequently used natural language processing toolkit, to conduct NER and assign POS tags for keywords. It can return the results of NER and tagger simultaneously by only loading model once. Moreover, in order to generate more diverse expressions, we enrich the word in templates with its synonyms from WordNet [65].

5. Evaluation

We conduct experiments on an image classification task and evaluate our framework regarding the following aspects: (1) accuracy of our attentive ZSL learner (AZSL) in the standard ZSL and generalized ZSL setting in comparison with the state-of-the-art ZSL baselines; (2) illustration of the feature transfer from seen classes to unseen classes; (3) evaluation on the generated explanations, including human scoring, qualitative analysis and case studies. Based on the generated explanations, we also discuss the transfer of deep features in ZSL.

5.1. Experiment Setting

5.1.1. Datasets

Two widely used image sets are adopted: Animals with Attributes (AwA) [6] and ImageNet [66]. AwA is a coarse-grained dataset, while ImageNet is diverse in terms of granularity, i.e., it contains a collection of fine-grained datasets, e.g., different vehicle types, as well as coarse-grained datasets. Each AwA class or 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 images while the latter do not but are semantically related to the former. Specifically, in ImageNet, 398 animal classes are used as seen classes, each of them contains about $1,000$ images, while the 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. It is noted that we only consider the "one-hop" unseen classes in ImageNet, although those more hops away can also be taken as unseen classes. It is because ZSL algorithms often perform worse when the unseen classes are far from the seen ones [3–5]. In order to investigate the explainable ZSL problem better, we focus on these one-hop classes which are visually and semantically similar with the seen classes.

[5]https://spacy.io/

We leverage WordNet to build the hierarchical graph of classes in our datasets. Specifically, we first make an alignment between ZSL classes and WordNet entities. And then, these classes are connected with each other via "subClassOf" relation edge, in our experiments, we adopt two strategies to construct the edge. One is to look up the hypernyms of classes using WordNet interface in NLTK toolkit[6]. The other is to utilize a publicly available hierarchical structure of all ImageNet classes[7], from which a substructure that includes the ZSL classes in our datasets is extracted as the hierarchical graph. These two strategies build the same class hierarchy for us, for more details please refer to our published code. Considering that AwA unseen classes are contained in the ImageNet unseen set and several of the seen classes (24 out of 40) overlap with the ImageNet seen set, we build a universal hierarchical graph for two datasets. The total number of graph nodes is $3,969$, in which the seen classes, unseen classes as well as their ancestors, descendants and siblings are connected with each other. Although AwA classes overlap with ImageNet classes, we train different AZSL models for different datasets considering their different granularity and data distribution.

Especially, as ImageNet contains not only coarse-grained subsets but also fine-grained subsets, the density of the connection between seen classes and unseen classes varies a lot: some seen (unseen) classes whose first-order neighbors contain multiple (e.g., 5) unseen (seen) classes (i.e., dense connection), while some seen (unseen) classes whose first-order neighbors are very few (e.g., 1) (i.e., sparse connection). Regarding different connection density, we extract a subset ImageNet* from ImageNet with all sparsely connected classes removed to evaluate our AZSL model, in which each seen (unseen) class is connected with more than two unseen (seen) classes. More statistics of the dataset are listed in Table 4.

Additionally, in our experiments, we take partial samples of seen classes as the validation set. Specifically, the images of each seen class are split into 2 parts – 80% of them are used for training and 20% are used for validation. For the generalized ZSL setting, where the testing set contains some testing images from seen classes, we further split the above validation images into 2 parts, i.e., 10% of images are still taken as validation set and 10% are taken as testing

[6]https://www.nltk.org/howto/wordnet.html

[7]http://www.image-net.org/api/xml/structure_released.xml

Table 4

Statistics of the image sets. "#Train/Val/Test" denotes the number of images for training/validation/testing. "Test(Seen/Unseen)" means the testing images of seen or unseen classes in the generalized ZSL setting. Notably, we only test on AwA and ImageNet in the generalized ZSL.

Dataset	Classes #	Seen Classes #	Unseen Classes #	# Train/Val/Test in standard ZSL	# Train/Val/Test (Seen/Unseen) in generalized ZSL
AwA	50	40	10	23,513 / 5,896 / 7,913	23,513 / 2,948 / 10,861 (2,948/7,913)
ImageNet	895	398	497	318,400 / 79,600 / 409,307	318,400 / 39,800/ 449,107 (39,800/409,307)
ImageNet*	473	174	299	139,200 / 34,800 / 244,925	- / - / -

set. More details of these splits are listed in Table 4. During validation, these validation images will be predicted on seen classes to evaluate the prediction ability of learned seen classifiers so as to ensure that we obtain well-trained unseen classifiers.

5.1.2. Baselines

The following ZSL methods are used as baselines: **DAP** and **IAP** [6] which utilize the image attributes to model the inter-class relationship, **DeViSE** [3] and **ConSE** [4] which linearly map image features into the class embedding space, **SYNC** [7] which develops a series of "phantom" classes as bases to associate seen and unseen classes in the class embedding space, **GCNZ** [5] and **DGP** [18] which utilize GCN and WordNet to learn classifiers for unseen classes.

In our AZSL, we leverage Attentive GCN to optimize the encoding of class knowledge so as to learn more effective unseen classifiers. To demonstrate the effectiveness of the attention layer in AGCN, we implement graph convolutional layers with the graph convolutional operations proposed in GCNZ and DGP. The model referring to GCNZ is denoted as **AZSL-G**, while referring to DGP is denoted as **AZSL-D**.

Besides, NIWT, which was proposed by Selvaraju et al. [9], is a work for explaining ZSL. However, in our paper, we do not make a comparison with it. It is because that NIWT focuses on justifying the predictions made by unseen classifiers and grounding the transferred neurons in interpretable semantics, while our work focuses on explaining the transferability of features from seen classes to unseen classes.

5.1.3. Model Configuration and Evaluation Metrics

We adopt ResNet50 – a successful CNN architecture to extract the features of images [67]. For ResNet50, the output parameter vector of the second to the last layer has $2,048$ dimensions, therefore, the dimensionality of the classifier in our paper is also set to $2,048$. Following GCNZ and DGP, we adopt 6 convolutional layers for AZSL-G and 2 convolutional layers for AZSL-D. Both of them contain one attention layer with an attention weight threshold $\alpha = 0.01$. The initial embeddings of class nodes (i.e., the word embeddings of class names) are trained on Wikipedia 2014 dump and Gigaword 5 corpus using Glove [68] model, whose dimension is set to 300. The activation function in graph convolutional layer is LeaklyReLU with negative input slope 0.2. We utilize validation set to tune the hyperparameters of AZSL. A grid search is conducted over parameter pools to explore the optimal ones, such as $\{0.0001, 0.0002, 0.005\}$ for the learning rate, $\{$5e-3, 5e-4, 5e-5$\}$ for the L_2 regularization. In the standard ZSL setting, the initial learning rate is finally set to 0.0002, the dropout parameter is set to 0.5, L_2 regularization parameter is set to 0.005, and Adam optimizer is adopted. The optimal hyperparameters in generalized ZSL are the same as in standard ZSL. For *EvidenceMining* algorithm, the minimum *support* and *confidence* value are set to 10% and 30% respectively.

We evaluate the ZSL model with **Hit@k** metric, which represents the percentage of samples whose top *k* scored labels hit the ground-truth label and is widely used for performance measurement in ZSL. Notably, in standard ZSL setting, the Hit@k is computed on the testing samples of unseen classes, while in generalized ZSL, the Hit@k is computed on the testing samples of seen and unseen classes separately, denoted as $\mathrm{Hit_s}$@k and $\mathrm{Hit_u}$@k respectively. We set *k* to 1, 2, 5 in the standard ZSL setting, and 1 in the generalized ZSL setting. $k = 1$ is widely believed to be the most important [17]. As AwA has only 10 unseen classes, we use Hit@1 (i.e., accuracy) alone in both two settings.

5.2. Evaluation of Attentive ZSL Learner

5.2.1. Standard ZSL Setting

We first report the results under the standard ZSL setting in Table 5. It can be seen that the performance of KG-based 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 are more effective in modeling the inter-class relationship and can significantly improve the ZSL performance. It is as ex-

Table 5

Performance (%) of AZSL-G, AZSL-D and baselines on AwA, ImageNet and ImageNet* in the standard ZSL setting. † indicates the results come from the original paper. "–" means the method cannot be applied to the dataset. "±" represents the variation range of results in the repeated experiments.

(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**

pected, because the semantics of class names and attributes used by traditional methods is not as rich as that of KG.

Compared with GCNZ and DGP – the state-of-the-art methods utilizing KG semantics, our AZSL-G and AZSL-D perform better in most 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 explanations for ZSL and it is widely believed that there is a compromise between a machine learning model's interpretation and accuracy [69], the performance improvement of AZSL over GCNZ and DGP is still very promising.

We also evaluate KG-based methods on ImageNet*, a dense graph we extract from ImageNet. We find that 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 AGCN in dealing with the densely connected KG. It also validates the assumption: the performance of ZSL model can be improved by taking the different contributions of different seen classes into consideration.

5.2.2. *Generalized ZSL Setting*

From the above results, we observe that the KG-based ZSL methods perform better than other traditional methods, therefore, in this subsection, we mainly report the prediction results of KG-based methods, as Table 6 shows. We find that the performance of all methods dramatically drops when predicting unseen testing samples (i.e., $Hit_u@1$) compared in the standard setting. It is expected because the label space contains both seen and unseen classes during testing and these models might tend to classify unseen testing samples as seen classes considering that they have never been trained with the samples of unseen classes.

Table 6

Performance (%) of AZSL-G, AZSL-D and KG-based baselines on AwA, ImageNet in the generalized ZSL setting.

Model	AwA		ImageNet	
	$Hit_s@1$	$Hit_u@1$	$Hit_s@1$	$Hit_u@1$
GCNZ	75.46	19.74	**50.53**	15.07
AZSL-G (us)	**76.41**	**24.44**	44.66	**15.67**
DGP	**78.85**	58.09	**56.03**	13.95
AZSL-D (us)	52.29	**65.54**	51.48	**15.30**

Table 7

Error analysis of DGP and AZSL-D on AwA in the generalized ZSL setting. "from Seen/Unseen" means the wrongly predicted labels are from seen class set or unseen class set.

Model	Misclassified Testing Samples	Ratio of Predicted Labels	
		from Seen (%)	from Unseen (%)
DGP	seen	82.6	17.4
	unseen	90.3	9.7
AZSL-D	seen	46.7	53.3
	unseen	75.5	24.5

To validate our assumption, we conduct error analysis on those wrongly classified unseen testing samples. As Table 7 shows, we count the distribution of predicted labels of these misclassified testing samples. Taking the prediction results of DGP on AwA as examples, 90.3% of all misclassified testing samples of unseen classes are wrongly predicted as seen classes, indicating the strong bias towards seen classes during prediction. While our method AZSL-D alleviates the bias – reducing the percentage of testing samples that are wrongly classified as seen classes (from 90.3% to 75.5%) and achieving 7.45% performance gain over DGP on AwA. Results of AZSL-G and performance on ImageNet also have similar trends.

We also find that our models perform not well when predicting seen testing samples (i.e., $Hit_s@1$) in most cases in comparison with baselines. It is likely to be because the models are more easily confused by the candidate classes from unseen class set during testing, according to the error analysis in Table 7. This motivates us to explore optimized algorithms to classify unseen testing samples correctly as well as retain high accuracy on seen testing samples.

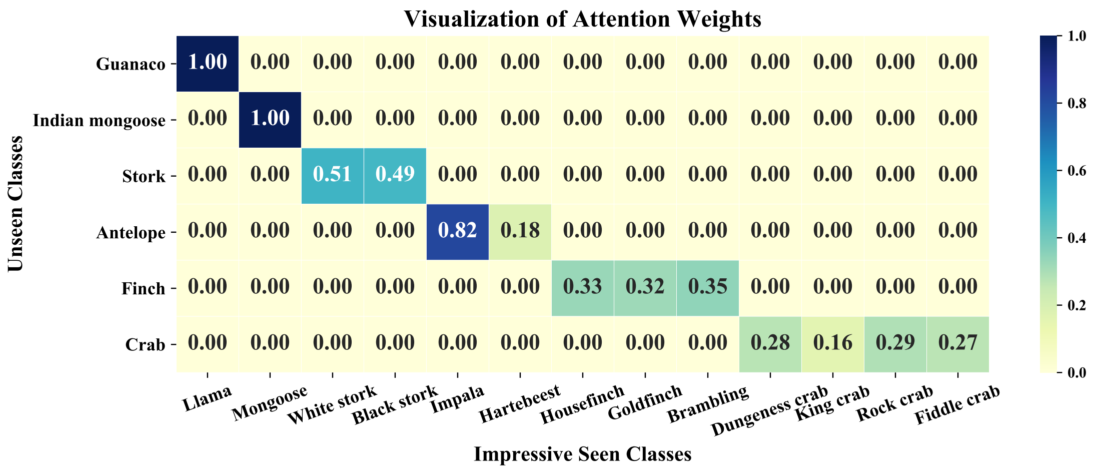


Fig. 6. Impressive seen classes (IMSCs) as well as their normalized attention weights of 6 randomly selected unseen classes. 0.00 here means a weight value below a threshold (very close to zero).

5.3. Illustration of Feature Transfer

In this subsection, we illustrate the transfer of deep features from seen classes to unseen classes with the learned impressive seen classes (IMSCs), including an intuitive visualization and some quantitative analyses.

Firstly, Figure 6 visualizes some unseen classes and their IMSCs, showing that these impressive seen classes transfer their deep features to the corresponding unseen classes. The presented examples in Figure 6 are mostly consistent with our common sense about animals, for example, in our impression, *Guanaco* and *Llama* are two animals that are similar in appearance. We also evaluate the impact of IMSCs by analyzing the performance drop when some IMSCs are removed, as shown in Figure 7. Taking the prediction results of AZSL-G as examples, the performance decreases in all cases when some IMSCs are removed, in comparison with NO removing. Specially, it drops to 0 in most cases when all IMSCs are removed. According to these observations, we can conclude that our AZSL is capable of learning reasonable IMSCs for unseen classes and these IMSCs play a key role in transferring their features to unseen classes. They can be used to generate explanations to analyze the transferability of features from seen classes to unseen classes.

We also observe that the number of IMSCs of different unseen classes varies dramatically. For example, *Indian mongoose* and *Guanaco* have only one IMSC, while *Finch* and *Crab* have 3 and 4 respectively. We further count the distribution of IMSC size in Table 8. An interesting finding is that most unseen classes

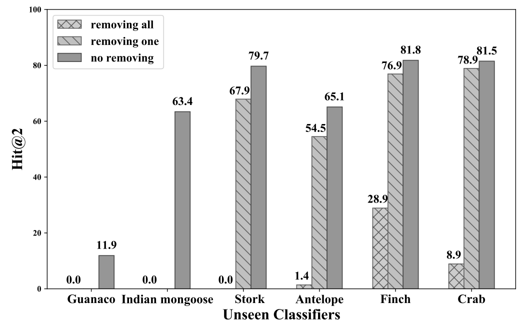


Fig. 7. Hit@2 of AZSL-G when one IMSC is removed, all IMSCs are removed and NO IMSCs are removed.

have only one IMSC (78.90%) or two IMSCs (5.72%), while around 2% of all unseen classes have more than two IMSCs. For example, class *Rat* has three impressive seen classes *Mouse*, *Hamster* and *Beaver*, each of them whose attributes extracted from Attribute Graph are more than 30. The searching space is large for finding a common attribute set among these classes. From our statistics, although the proportion of unseen classes with multiple IMSCs is not high, the *EvidenceMining* algorithm provides a demonstration for our system to be applied to other datasets and tasks. Moreover, we not only extract common attributes using the algorithm but also mine the association rules of seen and unseen classes with some measurements produced – *support* and *confidence* values, which illustrate the ratio of common attributes to all attributes of these classes.

We also note that some unseen classes (around 13.20%) do not have any impressive seen classes. It is probably because: *(i)* there are no neighboring seen

Table 8
The distribution of impressive seen classes (IMSCs).

Size of IMSC	0	1	2	3	4	5	6	7
Number of Unseen Class	67	400	29	5	3	1	1	1
Ratio (%)	13.20	78.90	5.72	0.99	0.59	0.20	0.20	0.20

Table 9
Results of human evaluation on the generated 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%

classes in the hierarchical graph, *(ii)* the visual features of neighboring seen classes are quite different from these unseen classes (meaning the gap between semantic domain and visual domain), or *(iii)* the disability of AZSL model, where the features can not be completely transferred from seen classes to these unseen classes.

5.4. Evaluation of Explanations

In this subsection, we demonstrate how human beings are satisfied with the generated textual 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. The first language of volunteers is Chinese, they are all undergraduate students who are fluent in reading English. We divide all unseen classes into 5 parts. Each one contains about 100 unseen classes and corresponding explanations. Each explanation is scored by 5 volunteers, 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 measures whether an explanation is natural and fluent. "Good": fluent, "Medium": unnatural but still understandable, "Bad": confused and incomprehensible.

Rationality measures whether an explanation illustrates the transferability of features between classes. "Good": well illustrated, "Medium": insufficient and weakly illustrated, "Bad": totally unconvincing.

These two metrics are scored independently. To help volunteers deal with the evaluation better, we prepare some guidelines and examples for them before scoring to make sure they are familiar with the scoring procedure. Besides, we also provide some images and textual illustrations of classes as references as well as some notes of generated explanations during scoring. It is allowed that volunteers can skip the scoring item if they are not sure about their judgment.

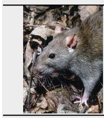
Table 9 presents the human evaluation results. We can find that the explanations of most unseen classes are satisfactory, especially on the rationality, and only a very small ratio of explanations get "Bad" on readability and rationality. It can also be seen that 60.77% of explanations get Median on readability, but they do not negatively impact 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 some examples of the generated explanations in Figure 8, with the evidence extracted from Attribute Graph and DBpedia. For the unseen class *Horse*, whose features are transferred from seen class *Zebra*, our explanation generator extracts attribute evidence such as *hooves, longneck, chewteech, tail* from Attribute Graph to illustrate the common attributes between *Horse* and *Zebra*, and mine the association rule: "$\{Zebra \Rightarrow Horse\}$" with *support* value 73.0% and *confidence* value 90.0%. As the upper left of Figure 8 shows. Also, the generator extracts general knowledge from DBpedia: some triples like *(dbr:Horse, dct:subject, dbr:Equus)* and *(dbr:Zebra, dct:subject, dbr:Equus)* which illustrate their common ancestor, and some keywords like *night vision* and *ears* which describe the same characteristics. With these evidence, we can justify the transfer of features from *Zebra* to *Horse* is reasonable (i.e., explain the transferability of features from *Zebra* to *Horse*).

We find that the evidence from different external KGs have different characteristics. Taking *Rat* in Figure 8 as an example, the evidence from Attribute Graph are more likely to generate explanations in vision, such as body parts (*quadrupedal, paws*) and coat appearances (*furry*), while the evidence from DBpedia usually generate more general explanations, including not only visual descriptions such as *incisors*, but also general descriptions such as *rodent* ancestor and *invasive mammal* in biology. We also have similar observations in other examples, this is due to the nature of these two

<table>
<tr><th colspan="2" rowspan="2"></th><th rowspan="2">Horse
(Unseen Class)</th><th>IMSCs of Horse</th></tr>
<tr><th>Zebra (w: 1.0)</th></tr>
<tr><td colspan="2">Image</td><td></td><td></td></tr>
<tr><td colspan="2">DBpedia Entity</td><td>dbr:Horse</td><td>dbr:Zebra</td></tr>
<tr><td colspan="2" rowspan="2">Knowledge from Attribute Graph</td><td colspan="2">hooves, longneck, chewteeth, tail, muscle, grazer, plains, grouped, quadrupedal, timid</td></tr>
<tr><td colspan="2">For rule $\{Zebra\} \Rightarrow \{Horse\}$:
sup. = 73.0%; con. = 90.0%</td></tr>
<tr><td rowspan="2">Knowledge from DBpedia</td><td>DBPedia Property/ Relation</td><td>(dbr:Horse, dct:subject, dbc:Equus)
(dbr:Horse, dct:subject, dbc: Herbivorous_animals)</td><td>(dbr:Zebra, dct:subject, dbc:Equus)
(dbr:Zebra, dct:subject, dbc:Herbivorous_animals)</td></tr>
<tr><td>Keywords of Abstract Text</td><td>ungulate mammal, domesticated, day and night vision, balance, large ears</td><td>black-and-white striped coats, ungulates, social, large ears, night vision</td></tr>
<tr><td colspan="2" rowspan="2">Generated Explanation & Score</td><td colspan="2">The prediction for samples of horse is supported by zebra. They are both grazer, grouped, quadrupedal, both have hooves, muscle, longneck, chewteeth, both live in plains, and behave timid. <u>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.</u></td></tr>
<tr><td colspan="2">Readability [G/M/B]: M
Rationality [G/M/B]: G</td></tr>
</table>

<table>
<tr><th rowspan="2">Rat
(Unseen Class)</th><th colspan="3">IMSCs of Rat</th></tr>
<tr><th>Mouse (w: 0.36)</th><th>Hamster(w:0.34)</th><th>Beaver (w: 0.30)</th></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>dbr:Rat</td><td>bdr:Mouse</td><td>dbr:Hamster</td><td>dbr:Beaver</td></tr>
<tr><td colspan="4">quadrupedal, ground, nocturnal, paws, small, buckteeth, hibernate, nestspot, agility, furry</td></tr>
<tr><td colspan="4">For rule $\{Mouse, Hamster,\ Beaver\} \Rightarrow \{Rat\}$:
sup. = 26.8%; con. = 88.2%</td></tr>
<tr><td colspan="4">(dbr:Rat, hypernym, dbr:Rodents)
(dbr:Hamster, hypernym, dbr:Rodents)
(dbr:Rat, dct:subject, dbc:Invasive_mammal)
(dbr:Mouse, dct:subject, dbc:Invasive_mammal)</td></tr>
<tr><td>medium-sized, long-tailed rodents, incisors</td><td>pointed incisors, rounded ears, Rodentia, high breeding rate</td><td>rodents, sharp incisors, house pets, underground,</td><td>semiaquatic rodent, nocturnal, building dams, four incisors</td></tr>
<tr><td colspan="4">The prediction for samples of rat is supported by mouse, hamster and beaver. They are both quadrupedal, nocturnal, small, furry, both have paws, buckteeth, both live in ground, nestspot, both hibernate and behave agility. <u>Rat and hamster are both relevant to rodents via relation hypernym, rat and mouse both belong to invasive mammal. They are both have incisors.</u></td></tr>
<tr><td colspan="4">Readability [G/M/B]: M
Rationality [G/M/B]: G</td></tr>
</table>

<table>
<tr><th colspan="2" rowspan="2"></th><th rowspan="2">Dolphin
(Unseen Class)</th><th>IMSCs of Dolphin</th></tr>
<tr><th>Killer whale (w: 1.0)</th></tr>
<tr><td colspan="2">Image</td><td></td><td></td></tr>
<tr><td colspan="2">DBpedia Entity</td><td>dbr:Dolphin</td><td>dbr:Killer_whale</td></tr>
<tr><td colspan="2" rowspan="2">Knowledge from Attribute Graph</td><td colspan="2">hairless, toughskin, flippers, swims, tail, ocean, grouped, smart, fast, active</td></tr>
<tr><td colspan="2">For rule $\{Killer\ whale\} \Rightarrow \{Dolphin\}$:
sup. = 48.6%; con. = 62.1%</td></tr>
<tr><td rowspan="2">Knowledge from DBpedia</td><td>DBpedia Property/ Relation</td><td>(dbr:Dolphin, dct:subject, dbc:Animals_that_use_echolocation)</td><td>(dbr:Killer_whale, dct:subject, dbc:Animals_that_use_echolocation)</td></tr>
<tr><td>Keywords of Abstract Text</td><td>aquatic, shaped teeth, well-developed hearing, widespread, blubber under skin</td><td>oceanic, apex predators, toothed whale, a layer of blubber, excellent hearing, diverse diet</td></tr>
<tr><td colspan="2" rowspan="2">Generated Explanation & Score</td><td colspan="2">The prediction for samples of dolphin is supported by killer whale. They are both hairless, grouped, both have tough skin, flippers, tail, both swim, live in ocean, and behave smart, fast and active. <u>They are both animals that use echolocation. They both have teeth, hearing and blubber.</u></td></tr>
<tr><td colspan="2">Readability [G/M/B]: M
Rationality [G/M/B]: M</td></tr>
</table>

<table>
<tr><th rowspan="2">Stork
(Unseen Class)</th><th colspan="2">IMSCs of Stork</th></tr>
<tr><th>White stork (w: 0.51)</th><th>Black stork (w: 0.49)</th></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>dbr:Ciconiiformes</td><td>dbr:White_stork</td><td>dbr:Black_stork</td></tr>
<tr><td colspan="3">white, black, water, wild, fish</td></tr>
<tr><td colspan="3">For rule $\{White\ stork, Black\ stork\} \Rightarrow \{Stork\}$:
sup. = 83.3%; con. = 100.0%</td></tr>
<tr><td colspan="3">(dbr:White_stork, dbo:order, dbr:Ciconiiformes)
(dbr:Black_stork, dbo:order, dbr:Ciconiiformes)</td></tr>
<tr><td>large, long-legged, long-necked, wading, birds, soaring, long, stout bills, migratory</td><td>large bird, long red legs, wading, family, Ciconiidae, migrant, carnivore</td><td>large bird, wading, family, Ciconiidae, black plumage, long red legs, red beak</td></tr>
<tr><td colspan="3">The prediction for samples of stork is supported by white stork and black stork. They are both white, black, wild, both live in water and eat fish. <u>White stork and black stork both belong to stork biologically. They are both large wadding birds, and are similar in long legs.</u></td></tr>
<tr><td colspan="3">Readability [G/M/B]: G
Rationality [G/M/B]: G</td></tr>
</table>

Fig. 8. The explanations for unseen classes: *Horse, Rat, Dolphin* and *Stork*. In each case, images of the unseen class and its impressive seen classes (IMSCs), their matched DBpedia entities, the extracted attributes, triples and keywords are displayed. The association rules of seen and unseen classes from *EvidenceMining* algorithm are listed with their measurements ("sup.": the *support* value, "con.": the *confidence* value). "(w:∗)" behind IMSC denotes the attention weights. The textual explanations as well as the evaluation results from volunteers are also displayed. The sentences in italic are explanations generated with knowledge from Attribute Graph (i.e., attributes), while those marked with underline are generated with knowledge from DBpedia (i.e., triples and keywords).

Table 10
The evaluation results of explanations with different numbers of common attributes.

Attributes #	Rationality			Readability		
	G	M	B	G	M	B
>10	85.71%	14.29%	0.00%	0.00%	100.00%	0.00%
$5 \sim 10$	75.00%	21.88%	3.12%	40.63%	56.25%	3.12%
<5	72.68%	20.36%	6.96%	37.37%	59.79%	2.84%

KGs – Attribute Graph is a domain-specific knowledge graph while DBpedia is a general one.

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, and a considerable portion of classes (about 90%), especially those from ImageNet, whose attributes extracted from Attribute Graph are no more than 10. For example, the attributes of *Stork* and its IMSCs are few (see the bottom right of Figure 8). In contrast, the resources and knowledge from DBpedia are abundant, which can be accessed as long as the ZSL classes are matched with DBpedia entities. This setting is very friendly for providing explanations for classes of large scale ZSL datasets. However, different from human-annotated attributes in Attribute Graph, the knowledge from DBpedia sometimes is noisy due to the natural expressions in keywords. For example, in the case of *Dolphin* and *Killer whale* (the bottom left of Figure 8), the keywords "well-developed hearing" and "excellent hearing" are extracted to describe their hearing, however, only "hearing" is taken as the common keywords due to the different adjectives. This incomplete knowledge may hurt the quality of generated explanations, motivating us to explore better keyword extraction algorithm to fully utilize the knowledge in abstract text.

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

5.4.3. Impact of Attribute Graph

From Figure 8, we observe that the number of attributes extracted for different classes from Attribute Graph varies a lot. For example, *Horse*, *Dolphin* and *Rat* both have 10 common attributes while *Stork* only has 5. In our statistics, 3.23% of all unseen classes (or explanations) have more than 10 common attributes extracted from Attribute Graph, 7.37% have $5 \sim 10$ common attributes while 89.4% have less than 5 attributes. Therefore, we further analyze the impact of the coverage of attributes on generating explanations. Specifically, we reevaluate the quality of generated explanations with different numbers of extracted common attributes, the results are shown in Table 10.

In all explanations with more than 10 common attributes, 85.71% of them are scored with "Good" on rationality, while 14.29% are with "Medium". The medium rationality may be because *(i)* we randomly select 10 attributes to generate explanations when the number of common attributes exceeds 10, however, some representative attributes especially those that are discriminative across different classes may not be selected, making the generation results less convincing, or *(ii)* the knowledge extracted from DBpedia may be not compelling enough. We also note that the proportion of explanations scored with "B" increases as the number of attributes decreases, indicating that the attributes can make up for the shortcomings of evidence from DBpedia. Besides, most explanations get "G" or "M" on rationality when the number of attributes is less than 10 or even less than 5, meaning that our system can still work well with DBpedia even though the attributes from Attribute Graph are not rich enough.

As for the quality in readability, we find that there is a slip when packing too many attributes. It might be because there are many repetitive expressions in the generated sentences. It is believed that randomly taking 10 attributes to generate is a good choice when the number of attributes is more than 10, however, some representative attributes may be lost as we mentioned above. Therefore, we look forward to adopting some strategies to improve the selection of attributes in the future, for example, evaluating the relevancy between classes and attributes to select the most relevant ones.

Generally speaking, the high coverage of attributes has a positive effect on generating higher quality explanations, especially in terms of rationality. However, it is better to combine the knowledge from Attribute Graph and DBpedia to complement each other.

5.5. Discussion on Feature Transfer in ZSL

In this subsection, we analyze the transfer of deep features in ZSL according to our generated explanations. We take the prediction results of AZSL-G in the standard ZSL setting as examples.

5.5.1. Successful and Failed Transfer

In ZSL, the 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 on Hit@1 and Hit@2 are both 0. Such a case is viewed as a *failed transfer* (FT). In contrast, the case where some unseen classes have features transferred from seen classes is a *successful transfer* (SF). For example, the Hit@1 and Hit@2 of unseen class *Eared seal* are both 0 and its feature transfer is failed, while another class *Frog* whose features are transferred from seen classes *Tree frog* and *Tailed frog* achieves 61.70% Hit@1 and 79.62% Hit@2, is a successful case.

Those explanations scored with "Good" on rationality illustrate the transferability of features from seen classes to unseen classes well, and are regarded as *good explanations*, while others are not. We find that 48% of 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 knowledge extracted from the KG, *(ii)* the incorrect or absent matching between classes and DBpedia entities as mentioned in Section 4.3.2, or *(iii)* the absence of reasonable impressive seen classes for unseen classes, while the successful transfer may be due to the semantics implied in the class embeddings (i.e., the initialization of class nodes). The first point can be solved by developing more advanced methods to extract knowledge, while the second point can be improved by traditional ontology alignment systems or modern semantic embedding methods. We also find around 3% of unseen classes are FT with good explanations. This may be because the learned unseen classifiers are not discriminative enough, resulting in poor performance on Hit@1 and Hit@2, or the feature extraction in CNN needs to be refined. This indicates that on the one hand, the class knowledge can be further enhanced to learn more discriminative unseen classifiers, on the other hand, the encoded class knowledge can be utilized to improve the CNN module. The rest (10%) of unseen classes are FT without good explanations. It may be because these unseen classes do not have related seen classes in the ZSL datasets, resulting in no seen features that can be transferred to them.

5.5.2. Different Types of Feature Transfer

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

Table 11

Performance of AZSL-G with different types of transferability.

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

We count all successful transfers in ImageNet according to the different types of feature transfer. As shown in Table 11, 49.2% of unseen classes, whose features are transferred from *ancestor* seen classes, achieve 25.06% on Hit@1, while 38.1% of unseen classes, whose features are transferred from *sibling* seen classes, achieve 29.10% on Hit@1. Nearly 90% of unseen classes focus on these two kinds of feature transfer. It is because the inter-class relationship our model input is hierarchical. We also find that the prediction results of unseen classes with *sibling* type are superior to those with *ancestor* type, probably because of the divergence of feature distribution between ancestor classes and descendant classes considering that the feature distribution of ancestor classes is more complex than that of descendant classes. It is inspired that the performance of ZSL model may be improved by introducing *sibling* type feature transfer (e.g., introducing more sibling seen classes for unseen classes). However, the combination of these two types achieves the best performance, indicating that richer class semantics are more helpful for the transfer of features.

6. Conclusion and Outlook

In this study, we investigate explainable ZSL with (1) a new ZSL learner which utilizes the inter-class relationships extracted from WordNet as well as the Attentive Graph Convolutional Network to predict classifiers for unseen classes, and (2) an explanation generator which generates human understandable explanations with external Attribute Graph and DBpedia to justify the transferability of features in ZSL. The study is evaluated with two image sets. We not only achieve higher classification performance than the state-of-the-art baselines, but also generate promising explanations which make the transferability of features in ZSL more explainable. With the generated explanations, we also analyze the transfer of features from seen classes to unseen classes, which shows the potential of further improving the performance of ZSL algorithms.

In our work, we extract class knowledge e.g., attributes, triples and keywords from Attribute Graph and DBpedia as evidence to illustrate the transferability of features from seen classes to unseen classes. From another point of view, these evidence also build detailed semantic relationships between classes, which may be helpful for the feature transfer in ZSL. Therefore, in the future work, we can integrate these common attributes, keywords and triples into the hierarchical graph of classes to enrich the class semantics and improve the performance of ZSL models [70, 71].

We also consider further improving the quality of explanations by making full use of the knowledge in external KGs. For example, we can use Semantic Web techniques such as ontology that involves the domain and range of properties to assist the knowledge extraction from Attribute Graph and DBpedia. Note that the ontology of DBpedia is ready-made, while the ontology of Attribute Graph may need to be designed manually considering it is collected from attribute annotations. We will explore this direction in the future.

Our work currently focuses on the image classification tasks, we also look forward to applying it in other domains like KG construction and natural language processing. For example, it can be applied for long-tail relation extraction [44] and zero-shot knowledge graph completion [72]. The models proposed in these tasks usually utilize label knowledge to transfer data features from seen (or data-rich) labels to unseen (or data-poor) labels, we can also introduce attention mechanism or other strategies into these models to attentively select the seen (or data-rich) labels which are contributing to the feature learning of unseen (or data-poor) labels. With learned contributing seen labels, the explanations of the feature transferability can be made by accessing the knowledge from external KGs as our work does. It is noted that the external KGs such as DBpedia can still be utilized to generate explanations for these tasks, while Attribute Graph is specific to image classification. If necessary, we can explore other external KGs for new applications or introduce other domain-specific knowledge from domain experts or public resources.

Acknowledgments

This work is funded by national key research program 2018YFB1402800, NSFC91846204/U19B2027, and supported by Alibaba-ZJU Frontier Technology Research Center. Jiaoyan Chen's contribution is supported by the AIDA project (Alan Turing Institute), the SIRIUS Centre for Scalable Data Access (Research Council of Norway), Samsung Research UK, Siemens AG, and the EPSRC projects AnaLOG (EP/P025943/1), OASIS (EP/S032347/1) and UK FIRES (EP/S019111/1).

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. doi:10.1007/s11263-015-0816-y.

[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 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams and A. Culotta, eds, Curran Associates, Inc., 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 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C.J.C. Burges, L. Bottou, Z. Ghahramani and K.Q. Weinberger, eds, 2013, pp. 2121–2129.

[4] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. Corrado and J. Dean, Zero-Shot Learning by Convex Combination of Semantic Embeddings, in: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, eds, 2014.

[5] X. Wang, Y. Ye and A. Gupta, Zero-Shot Recognition via Semantic Embeddings and Knowledge Graphs, in: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 6857–6866. doi:10.1109/CVPR.2018.00717.

[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. doi:10.1109/TPAMI.2013.140.

[7] S. Changpinyo, W.-L. Chao, B. Gong and F. Sha, Synthesized classifiers for zero-shot learning, in: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 5327–5336. doi:10.1109/CVPR.2016.575.

[8] E. Kodirov, T. Xiang and S. Gong, Semantic autoencoder for zero-shot learning, in: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 4447–4456. doi:10.1109/CVPR.2017.473.

[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: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, V. Ferrari, M. Hebert, C. Sminchisescu and Y. Weiss, eds, Lecture Notes in Computer Science, Vol. 11217, Springer, 2018, pp. 540–556. doi:10.1007/978-3-030-01261-8_32.

[10] H. Larochelle, D. Erhan and Y. Bengio, Zero-data Learning of New Tasks, in: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, D. Fox and C.P. Gomes, eds, AAAI Press, 2008, pp. 646–651.

[11] C.H. Lampert, H. Nickisch and S. Harmeling, Learning to detect unseen object classes by between-class attribute transfer, in: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, IEEE Computer Society, 2009, pp. 951–958. doi:10.1109/CVPR.2009.5206594.

[12] A. Farhadi, I. Endres, D. Hoiem and D. Forsyth, Describing objects by their attributes, in: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, IEEE Computer Society, 2009, pp. 1778–1785. doi:10.1109/CVPR.2009.5206772.

[13] Z. Akata, F. Perronnin, Z. Harchaoui and C. Schmid, Label-embedding for attribute-based classification, in: *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, IEEE Computer Society, 2013, pp. 819–826. doi:10.1109/CVPR.2013.111.

[14] P. Morgado and N. Vasconcelos, Semantically consistent regularization for zero-shot recognition, in: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 2037–2046. doi:10.1109/CVPR.2017.220.

[15] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein and B. Schiele, Latent embeddings for zero-shot classification, in: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, pp. 69–77. doi:10.1109/CVPR.2016.15.

[16] L. Zhang, T. Xiang and S. Gong, Learning a Deep Embedding Model for Zero-Shot Learning, in: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 3010–3019. doi:10.1109/CVPR.2017.321.

[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* **41**(9) (2018), 2251–2265. doi:10.1109/TPAMI.2018.2857768.

[18] M. Kampffmeyer, Y. Chen, X. Liang, H. Wang, Y. Zhang and E.P. Xing, Rethinking Knowledge Graph Propagation for Zero-Shot Learning, in: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 11487–11496. doi:10.1109/CVPR.2019.01175.

[19] P.K. Pushp and M.M. Srivastava, Train Once, Test Anywhere: Zero-Shot Learning for Text Classification, *CoRR* **abs/1712.05972** (2017). http://arxiv.org/abs/1712.05972.

[20] 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran and T. Solorio, eds, Association for Computational Linguistics, 2019, pp. 1031–1040. doi:10.18653/v1/n19-1108.

[21] Z. Ye, Y. Geng, J. Chen, J. Chen, X. Xu, S. Zheng, F. Wang, J. Zhang and H. Chen, Zero-shot Text Classification via Reinforced Self-training, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter and J.R. Tetreault, eds, Association for Computational Linguistics, 2020, pp. 3014–3024.

[22] L. Logeswaran, M. Chang, K. Lee, K. Toutanova, J. Devlin and H. Lee, Zero-Shot Entity Linking by Reading Entity Descriptions, in: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, A. Korhonen, D.R. Traum and L. Màrquez, eds, Association for Computational Linguistics, 2019, pp. 3449–3460. doi:10.18653/v1/p19-1335.

[23] S. Rijhwani, J. Xie, G. Neubig and J. Carbonell, Zero-shot neural transfer for cross-lingual entity linking, in: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, AAAI Press, 2019, pp. 6924–6931. doi:10.1609/aaai.v33i01.33016924.

[24] 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), Vancouver, Canada, August 3-4, 2017*, R. Levy and L. Specia, eds, Association for Computational Linguistics, 2017, pp. 333–342. doi:10.18653/v1/K17-1034.

[25] M. Johnson, M. Schuster, Q.V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado et al., Google's multilingual neural machine translation system: Enabling zero-shot translation, *Transactions of the Association for Computational Linguistics* **5** (2017), 339–351.

[26] O. Biran and C. Cotton, Explanation and justification in machine learning: A survey, in: *International Joint Conference on Artificial Intelligence (IJCAI) workshop on Explainable AI (XAI)*, 2017, p. 8.

[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. doi:10.1145/3236009.

[28] Z.C. Lipton, The Mythos of Model Interpretability, *ACM Queue* **16**(3) (2018), 30. doi:10.1145/3236386.3241340.

[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, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A.J. Smola, C.C. Aggarwal, D. Shen and R. Rastogi, eds, ACM, 2016, pp. 1135–1144. doi:10.1145/2939672.2939778.

[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: *IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, Singapore, 16-19 April, 2013*, IEEE, 2013, pp. 32–38. doi:10.1109/CIDM.2013.6597214.

[32] T. Kulesza, M. Burnett, W.-K. Wong and S. Stumpf, Principles of explanatory debugging to personalize interactive machine learning, in: *Proceedings of the 20th International Conference on Intelligent User Interfaces, IUI 2015, Atlanta, GA, USA, March 29 - April 01, 2015*, O. Brdiczka, P. Chau, G. Carenini, S. Pan and P.O. Kristensson, eds, ACM, 2015, pp. 126–137. doi:10.1145/2678025.2701399.

[33] D.S. Weld and G. Bansal, The challenge of crafting intelligible intelligence, *Communications of the ACM* **62**(6) (2019), 70–79. doi:10.1145/3282486.

[34] 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, Sydney, NSW, Australia, August 10-13, 2015*, L. Cao, C. Zhang, T. Joachims, G.I. Webb, D.D. Margineantu and G. Williams, eds, ACM, 2015, pp. 1721–1730. doi:10.1145/2783258.2788613.

[35] O. Biran and K.R. McKeown, Human-Centric Justification of Machine Learning Predictions, in: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, C. Sierra, ed., ijcai.org, 2017, pp. 1461–1467. doi:10.24963/ijcai.2017/202.

[36] Q. Li, J. Fu, D. Yu, T. Mei and J. Luo, Tell-and-Answer: Towards Explainable Visual Question Answering using Attributes and Captions, in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, E. Riloff, D. Chiang, J. Hockenmaier and J. Tsujii, eds, Association for Computational Linguistics, 2018, pp. 1338–1346. doi:10.18653/v1/d18-1164.

[37] D. Ratcliffe and K.L. Taylor, Closed-World Concept Induction for Learning in OWL Knowledge Bases, in: *Knowledge Engineering and Knowledge Management - 19th International Conference, EKAW 2014, Linköping, Sweden, November 24-28, 2014. Proceedings*, K. Janowicz, S. Schlobach, P. Lambrix and E. Hyvönen, eds, Lecture Notes in Computer Science, Vol. 8876, Springer, 2014, pp. 429–440. doi:10.1007/978-3-319-13704-9_33.

[38] I. Tiddi, M. d'Aquin and E. Motta, Dedalo: Looking for clusters explanations in a labyrinth of linked data, in: *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab and A. Tordai, eds, Lecture Notes in Computer Science, Vol. 8465, Springer, 2014, pp. 333–348. doi:10.1007/978-3-319-07443-6_23.

[39] J. Chen, F. Lécué, J.Z. Pan, I. Horrocks and H. Chen, Knowledge-based Transfer Learning Explanation, in: *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*, M. Thielscher, F. Toni and F. Wolter, eds, AAAI Press, 2018, pp. 349–358.

[40] Z. Yang, D. Yang, C. Dyer, X. He, A.J. Smola and E.H. Hovy, Hierarchical Attention Networks for Document Classification, in: *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, K. Knight, A. Nenkova and O. Rambow, eds, The Association for Computational Linguistics, 2016, pp. 1480–1489. doi:10.18653/v1/n16-1174.

[41] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang and G.W. Cottrell, A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction, in: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, C. Sierra, ed., ijcai.org, 2017, pp. 2627–2633. doi:10.24963/ijcai.2017/366.

[42] S.J. Pan and Q. Yang, A survey on transfer learning, *IEEE Transactions on knowledge and data engineering* **22**(10) (2009), 1345–1359. doi:10.1109/TKDE.2009.191.

[43] K. Weiss, T.M. Khoshgoftaar and D. Wang, A survey of transfer learning, *Journal of Big data* **3**(1) (2016), 9. doi:10.1186/s40537-016-0043-6.

[44] 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, in: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran and T. Solorio, eds, Association for Computational Linguistics, 2019, pp. 3016–3025. doi:10.18653/v1/n19-1306.

[45] C. Lee, W. Fang, C. Yeh and Y.F. Wang, Multi-Label Zero-Shot Learning With Structured Knowledge Graphs, in: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, IEEE Computer Society, 2018, pp. 1576–1585. doi:10.1109/CVPR.2018.00170.

[46] F. Lécué, J. Chen, J.Z. Pan and H. Chen, Augmenting Transfer Learning with Semantic Reasoning, in: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-*

16, 2019, S. Kraus, ed., ijcai.org, 2019, pp. 1779–1785. doi:10.24963/ijcai.2019/246.

[47] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, How transferable are features in deep neural networks?, in: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence and K.Q. Weinberger, eds, 2014, pp. 3320–3328.

[48] T. Liu, Q. Yang and D. Tao, Understanding How Feature Structure Transfers in Transfer Learning, in: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, C. Sierra, ed., ijcai.org, 2017, pp. 2365–2371. doi:10.24963/ijcai.2017/329.

[49] Y. Geng, J. Chen, E. Jiménez-Ruiz and H. Chen, Human-centric Transfer Learning Explanation via Knowledge Graph [Extended Abstract], *CoRR* **abs/1901.08547** (2019). http://arxiv.org/abs/1901.08547.

[50] G.A. Miller, WordNet: a lexical database for English, *Communications of the ACM* **38**(11) (1995), 39–41. doi:10.1145/219717.219748.

[51] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie and P. Perona, Caltech-UCSD Birds 200, Technical Report, CNS-TR-2010-001, California Institute of Technology, 2010.

[52] 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. doi:10.3233/SW-140134.

[53] T.N. Kipf and M. Welling, Semi-Supervised Classification with Graph Convolutional Networks, in: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[54] 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 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C.J.C. Burges, L. Bottou, Z. Ghahramani and K.Q. Weinberger, eds, 2013, pp. 3111–3119.

[55] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio, Graph Attention Networks, *CoRR* **abs/1710.10903** (2017). http://arxiv.org/abs/1710.10903.

[56] R. Agrawal, T. Imieliński and A. Swami, Mining association rules between sets of items in large databases, in: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993*, Vol. 22, P. Buneman and S. Jajodia, eds, ACM Press, 1993, pp. 207–216. doi:10.1145/170035.170072.

[57] R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules in Large Databases, in: *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, J.B. Bocca, M. Jarke and C. Zaniolo, eds, Morgan Kaufmann, 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 the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, C. Ghidini, A.N. Ngomo, S.N. Lindstaedt and T. Pellegrini, eds, ACM International Conference Proceeding Series, ACM, 2011, pp. 1–8. doi:10.1145/2063518.2063519.

[59] A.N. Ngomo and S. Auer, LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data, in: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, T. Walsh, ed., IJCAI/AAAI, 2011, pp. 2312–2317. doi:10.5591/978-1-57735-516-8/IJCAI11-385.

[60] M. Pershina, M. Yakout and K. Chakrabarti, Holistic entity matching across knowledge graphs, in: *2015 IEEE International Conference on Big Data, Big Data 2015, Santa Clara, CA, USA, October 29 - November 1, 2015*, IEEE Computer Society, 2015, pp. 1585–1590. doi:10.1109/BigData.2015.7363924.

[61] Q. Zhang, Z. Sun, W. Hu, M. Chen, L. Guo and Y. Qu, Multi-view Knowledge Graph Embedding for Entity Alignment, in: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, S. Kraus, ed., ijcai.org, 2019, pp. 5429–5435. doi:10.24963/ijcai.2019/754.

[62] Y. Wu, X. Liu, Y. Feng, Z. Wang and D. Zhao, Jointly Learning Entity and Relation Representations for Entity Alignment, 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, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng and X. Wan, eds, Association for Computational Linguistics, 2019, pp. 240–249. doi:10.18653/v1/D19-1023.

[63] R. Mihalcea and P. Tarau, Textrank: Bringing order into text, in: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, ACL, 2004, pp. 404–411.

[64] D. Nadeau and S. Sekine, A survey of named entity recognition and classification, *Lingvisticae Investigationes* **30**(1) (2007), 3–26. doi:10.1162/COLI_a_00178.

[65] 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.

[66] J. Deng, W. Dong, R. Socher, L. Li, K. Li and F. Li, ImageNet: A large-scale hierarchical image database, in: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, IEEE Computer Society, 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848.

[67] K. He, X. Zhang, S. Ren and J. Sun, Deep Residual Learning for Image Recognition, in: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.

[68] 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 Pro-*

cessing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, A. Moschitti, B. Pang and W. Daelemans, eds, ACL, 2014, pp. 1532–1543. doi:10.3115/v1/d14-1162.

[69] H. Paulheim, Make Embeddings Semantic Again!, in: *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018*, M. van Erp, M. Atre, V. López, K. Srinivas and C. Fortuna, eds, CEUR Workshop Proceedings, Vol. 2180, CEUR-WS.org, 2018.

[70] Y. Geng, J. Chen, Z. Chen, Z. Ye, Z. Yuan, Y. Jia and H. Chen, Generative Adversarial Zero-shot Learning via Knowledge Graphs, *CoRR* **abs/2004.03109** (2020). https://arxiv.org/abs/2004.03109.

[71] J. Chen, F. Lécué, Y. Geng, J.Z. Pan and H. Chen, Ontology-guided Semantic Composition for Zero-shot Learning, in: *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, D. Calvanese, E. Erdem and M. Thielscher, eds, 2020, pp. 850–854. doi:10.24963/kr.2020/87.

[72] P. Qin, X. Wang, W. Chen, C. Zhang, W. Xu and W.Y. Wang, Generative Adversarial Zero-Shot Relational Learning for Knowledge Graphs, in: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, AAAI Press, 2020, pp. 8673–8680.