Identifying Entity Properties from Text with Zero-shot Learning

Wiradee Imrattanatrai
Kyoto University
wiradee@db.soc.i.kyoto-u.ac.jp

Makoto P. Kato
University of Tsukuba
JST, PRESTO
mpkato@slis.tsukuba.ac.jp

Masatoshi Yoshikawa
Kyoto University
yoshikawa@i.kyoto-u.ac.jp

ABSTRACT

We propose a method for identifying a set of entity properties from text. Identifying entity properties is similar to a relation extraction task that can be cast as a classification of sentences. Normally, this task can be achieved by distant supervised learning by automatically preparing training sentences for each property; however, it is impractical to prepare training sentences for every property. Therefore, we describe a zero-shot learning problem for this task and propose a neural network-based model that does not rely on a complete training set comprising training sentences for every property. To achieve this, we utilize embeddings of properties obtained from a knowledge graph embedding using different components of a knowledge graph structure. The embeddings of properties are combined with the model to enable identification of properties with no available training sentences. By using our newly constructed dataset as well as an existing dataset, experiments revealed that our model achieved a better performance for properties with no training sentences, relative to baseline results, even comparable to that achieved for properties with training sentences.

KEYWORDS

Sentence classification; property identification; property embedding

ACM Reference Format:
Wiradee Imrattanatrai, Makoto P. Kato, and Masatoshi Yoshikawa. 2019. Identifying Entity Properties from Text with Zero-shot Learning. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3331184.3331220

1 INTRODUCTION

Entities play an important role in modern search engines due to the significant number of entity-related queries [7, 19, 32] and a rapidly growing number of entities over the Web. To enable a more efficient entity-centric search, new search functionalities have been introduced, a successful example of which is *entity card*, which directly presents entity information, such as a short description and a set of facts, in conjunction with organic search results. The entity card has been proven as engaging and useful when it contains entity information relevant to the information needs of users [2].

To provide entity information relevant to the user's information need, an important task involves identifying *entity properties*, such as **album** and **genre** of the entity Taylor Swift and **school type** and **campuses** of the entity Stanford University, from textual information, including documents or sentences. For example, given the text *"2012 album Red took Taylor Swift's popularity to new levels and the universal appeal of I Knew You Were Trouble was a key part of that success."*, the task is to identify the property **album**, as well as **track**, of the entity Taylor Swift.

Identifying entity properties enables efficient access to relevant entity information. We can infer relevant entity properties according to the documents clicked by the user or those at the top rank of the search results, which can be used for composing a more relevant entity card, and offering additional relevant documents if those relevant entity properties are identified. For example, if a user clicks on documents describing entity properties **album** and **track** of the entity Taylor Swift, we can estimate that he/she is interested in these two properties, thereby enabling us to offer an entity card including albums and tracks and to suggest other documents describing these two properties. In addition, it is also possible to present properties described in each document directly on the initial search engine result page, to enable efficient navigation to relevant entity information. Suppose that a user issues the query, "Taylor Swift career highlights", directly presenting the properties described in each document helps the user efficiently select the relevant documents. Therefore, identifying entity properties from text is considered as a fundamental task in entity-centric searches, which can be applied to several practical applications.

In this study, we addressed the problem of identifying entity properties from text. This task is similar to relation extraction, where one of the most promising approaches involves supervised learning through distant supervision [10, 17, 22, 27]. In this case, the training sentences are obtained by utilizing triplets of the property in an existing knowledge base (KB). Some properties correspond to a single predicate (*e.g.* the property **music.artist.album** of the entity Taylor Swift corresponds to the predicate *music.artist.album*), and sentences are used as training examples if they contain the subject (e.g. Taylor Swift) and object (e.g. her album name) of triplets of the predicate. On the other hand, the other properties correspond to a *predicate path*, a set of connected predicates, such as the property **music.artist.track_contributions–music.track_contribution.track** of the entity Taylor Swift corresponds to the predicate path consisting of the predicates *music.artist.track_contributions* and *music.track_contribution.track*. Thus, for this type of properties, a set of triplets including their predicate path is used to identify training sentences. This distant supervision approach is efficient for preparing many sentences without requiring manual annotation.

However, since obtaining the training sentences relies heavily upon the triplets in the KB, a few or none of the sentences representing a certain property can be obtained if the predicate path representing the property is associated with a small number of triplets. Importantly, there exist hundreds of thousands of predicate paths that could be properties requiring identification. Thus, it is impractical to prepare training sentences for every property. A lack of training sentences for some properties results in our inability to learn the text pattern in order to classify those unseen properties (*i.e.* the properties without training sentences) in the learning model. This problem is well-known as the zero-shot learning (ZSL) problem, which to the best of our knowledge, has not previously been addressed in this context. Being unable to identify certain properties could result in low coverage of properties found in sentences, and more importantly, result in the users not acquiring important information concerning the entity.

To identify properties in sentences, we propose a neural network-based model for multi-label sentence classification. The model utilizes the type of the entity, as well as the property embeddings constructed based on the knowledge graph embedding (*i.e.* TransE [1]). We utilized the property embeddings to handle the ZSL problem. Based on the knowledge graph embedding, we propose several techniques to construct property embeddings, with each technique using different components in the knowledge graph, as the property needs to be well-represented by the embeddings to ensure that the model can effectively learn to classify both seen and unseen properties. In the experiment, we used our newly constructed sentence classification dataset, as well as an existing dataset, to compare performances between variations of our proposed model and baseline methods. We found that our model was able to identify more properties represented in sentences as compared with baseline methods. Additionally, our model was able to handle and predict properties unassociated with any sentences during model training, and its performance was comparable to that achieved for properties with training sentences.

In summary, the contributions of this study are three-fold:

- We introduced a neural network-based model for multi-label sentence classification to identify properties represented in sentences.
- We addressed the ZSL problem by extending the proposed model to utilize property embeddings constructed based on a knowledge graph embedding.
- We built a new sentence classification dataset comprising a larger number of properties to evaluate our proposed model along with an existing dataset.

2 RELATED WORK

In this section, we review existing studies concerning relation extraction and the ZSL problem.

Relation extraction can be cast as a sentence classification task, where the core problem involves classification of given sentences representing different properties. The most traditional approach applies supervised learning to a set of human-labeled sentences [4, 5, 8, 18, 33]. However, producing a large set of labeled sentences based solely on human judgment is limited and troublesome. To address this limitation, distant supervision was introduced and has become a promising technique adopted by most recent relation extraction works (*e.g.* [10, 17, 22, 27]). For distant supervision, sentences are automatically aligned with the corresponding properties based on triplets in an existing KB. Mintz *et al.* [17] assumed that if any sentences in a text corpus contain the subject and object of the triplet, they somehow contribute to the property of the predicate path of that triplet. According to this assumption, the sentences representing each property might be sufficiently obtained; however, this assumption is too strong because the sentence does not always represent the property, even if it contains a particular subject and object pair. Previous studies [3, 10, 22, 27] addressed this problem by applying multi-instance learning with a more relaxed assumption that at least one sentence containing the subject and object of the triplet represents the property. Additionally, other studies attempted to reduce the chance of producing incorrectly labeled sentences. Takamatsu *et al.* [28] proposed a generative model that models the probabilities of sentence patterns for the properties and uses negative sentence patterns to eliminate incorrectly labeled sentences. Xu *et al.* [30] presented a passage retrieval model that provides relevance feedback for filling in missing triplets. Similar to [30], Min *et al.* [16] and Ritter *et al.* [23] addressed the problem of an incomplete KB. Min *et al.* [16] utilized a semi-supervised approach that models true bag-level labels as the extension to [27], whereas Ritter *et al.* [23] introduced a new latent-variable approach that models missing triplets as the extension to [10].

While the problem of incorrectly labeled sentences has been addressed previously, their approaches do not specifically handle many types of objects in the KB. Because most works focused on extracting sentences with a subject and object pair that are also the entity names, the properties with triplets comprising an object that is a non-entity (*i.e.* a numerical value) are simply ignored. This kind of property is more likely to be connected to a higher number of incorrectly labeled sentences and could result in a zero sentence after applying existing techniques for penalizing these sentences.

In addition, another line of study on relation extraction applied the neural network-based model for classifying entity pairs in sentences as representing certain properties [26, 31, 34–37]; however, the models relied on a complete training set comprising of training sentences for every property. As the existing models were not specifically designed to handle the situation where the training set is incomplete, these models might not be able to identify properties with no training sentences. A lack of training examples for some classes in supervised learning is known as the ZSL problem.

ZSL has gained attention in the computer vision community, especially concerning image or object classification tasks. Recent ZSL studies utilized semantic embeddings of seen and unseen classes modeled using external information, such as class attributes [12], word vectors [6, 25, 29] and text description [13, 21]. This embedding model is used simultaneously with a classifier to enable prediction of classes with no available examples during training.

To address the problem of ZSL in our work, we used class embeddings and considered different techniques for constructing property embeddings by relying on the knowledge graph embedding which is achieved by a well-known technique (*i.e.* TransE [1]).

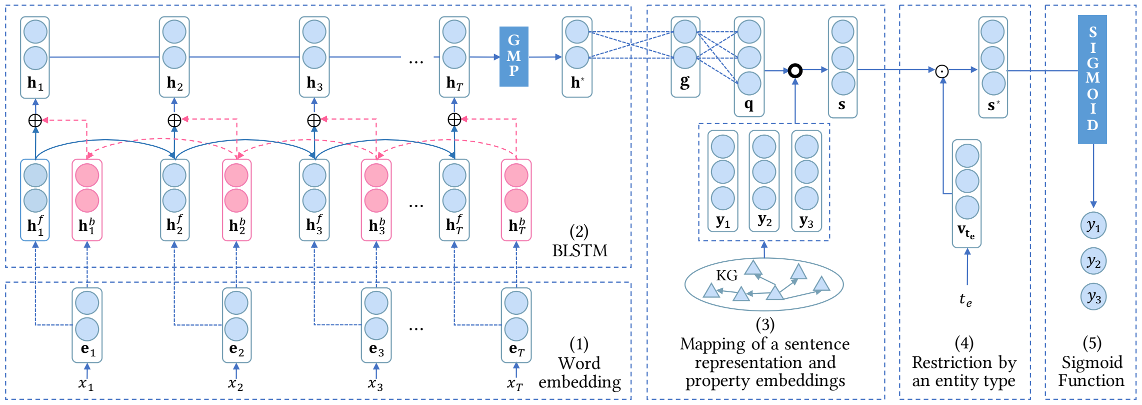

GMP
SIGMOID
KG
(1) Word embedding
(2) BLSTM
(3) Mapping of a sentence representation and property embeddings
(4) Restriction by an entity type
(5) Sigmoid Function

Figure 1: Model architecture.

3 METHODOLOGY

In this section, we present the neural network-based model for multi-label sentence classification. The model utilizes an entity type for narrowing down potential properties represented in sentences and property embeddings constructed based on a knowledge graph embedding for handling a ZSL problem.

3.1 Problem Definition

Let a set of properties Y be a set of classes into which we want to classify sentences. Given a target entity e in a given sentence X for which the model identifies the properties, our problem involves measuring the probability of each property, $y \in Y$ based on how certain each property is represented in the sentence X. At training time, a set of training sentences for some properties, $Y_{\text{seen}} \subset Y$, is used. At test time, a set of sentences for both seen properties, Y_{seen}, and unseen properties, Y_{unseen}, where $Y_{\text{unseen}} \cap Y_{\text{seen}} = \emptyset$, is classified. Therefore, according to the ZSL scenario, the model learns to classify the sentences based on the properties, Y_{train}, where $Y_{\text{train}} = Y_{\text{seen}}$ during training while the model predicts Y_{test}, where $Y_{\text{test}} = Y_{\text{seen}} \cup Y_{\text{unseen}}$ during testing.

3.2 Model

Our proposed model comprises five main components (Figure 1): (1) a word embedding that transforms each word in an input sentence into a low-dimensional word vector, (2) a bidirectional long short-term memory (BLSTM) that models a sequence of words and produces a sentence-level representation, (3) a mapping that learns the transformation of sentence representations with the embedding of properties based on a knowledge graph, (4) an entity type component that filters out properties that are definitely not represented by the input sentence, and (5) a sigmoid function that produces probabilities over all properties.

The goal is to learn how to map representations of sentences according to the embeddings of properties using the transformation matrix in order to ensure that sentence representation is similar to the embeddings of properties represented in the sentence. This allows the model to handle and predict any unseen properties represented by the sentence at test time.

3.2.1 Word embedding. The first component maps each word in the input sentence to the word vector that captures the semantic meaning of the word. Given a sentence comprising T words, $X = \{x_1, x_2, \ldots, x_T\}$, every word, x_t, is represented by a one-hot representation, $\mathbf{v}_t$ of size $|V|$, where the value at i-*th* is set to 0, except for an index of the word x_t, which is set to 1, and V is a fixed-size vocabulary. Each $\mathbf{v}_t$ is converted into a word vector, $\mathbf{e}_t$ by looking up the word embedding matrix, $\mathbf{E_w} \in \mathbb{R}^{d_e \times |V|}$, where d_e is the word embedding dimension, as $\mathbf{e}_t = \mathbf{E_w}\mathbf{v}_t$.

3.2.2 BLSTM. We chose to apply BLSTM to model a sequence of words in a sentence. An LSTM is a special type of recurrent neural network designed to handle a long-term dependency problem on the input sequences [9]. Given a sequence of word embeddings, $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_T\}$, for a sentence, X, each word embedding, $\mathbf{e}_t$, and the previous hidden output, $\mathbf{h}_{t-1}$, are passed to the LSTM unit at each step, t. The hidden output, $\mathbf{h}_t$, of each LSTM unit can be derived as follows:

$$\mathbf{i}_t = \sigma\left(\mathbf{W_i}\left[\mathbf{h}_{t-1}, \mathbf{e}_t\right] + \mathbf{b_i}\right) \tag{1}$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W_f}\left[\mathbf{h}_{t-1}, \mathbf{e}_t\right] + \mathbf{b_f}\right) \tag{2}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W_o}\left[\mathbf{h}_{t-1}, \mathbf{e}_t\right] + \mathbf{b_o}\right) \tag{3}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{W_c}\left[\mathbf{h}_{t-1}, \mathbf{e}_t\right] + \mathbf{b_c}\right) \tag{4}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right) \tag{5}$$

where σ denotes a sigmoid function, tanh denotes a hyperbolic tangent function, $\odot$ denotes an element-wise multiplication, $\mathbf{W_i}$, $\mathbf{W_f}$, $\mathbf{W_o}$, $\mathbf{W_c} \in \mathbb{R}^{d_h \times (d_e + d_h)}$ are weight matrices, where d_e and d_h denote the word embedding and hidden output dimensions, and $\mathbf{b_i}$, $\mathbf{b_f}$, $\mathbf{b_o}$, $\mathbf{b_c}$ are the biases for the input gate, forget gate, output gate, and cell-memory state, respectively.

At each step of the unidirectional recurrent network, the model learns to preserve information from previous steps as the input sequence is fed in chronological order. Consequently, the model cannot utilize the information from future steps to effectively learn the context of the word at the current step. To increase the effectiveness of modeling the input sequences, a bidirectional recurrent network was introduced [24], which added another recurrent layer to enable modeling of the input sequence in two directions (*i.e.* forward and backward). In this study, we used a BLSTM, with the hidden output at each step, t, obtained by element-wise addition between the hidden output from the forward ($\mathbf{h}_t^f$) and backward ($\mathbf{h}_t^b$) layers as $\mathbf{h}_t = \mathbf{h}_t^f \oplus \mathbf{h}_t^b$.

At the end of this component, we apply global max pooling (GMP) to every hidden output, $\mathbf{h}_t$, such that the model pays attention to the significant word of each feature. We achieved sentence representation as $\mathbf{h}^* = \max_t \{\mathbf{h}_t(i)\}$, where i indicates each index in the hidden output.

3.2.3 Mapping between property embedding and sentence representation. The third component in our model addresses the ZSL situation by learning the mapping between embeddings of properties and the sentence representation. Note that the construction of property embeddings will be explained in Section 3.4. To enable the model to learn to identify properties, as well as those not presented by any training sentences, we first transform the output from the previous component using a fully connected neural network with non-linear activation (*i.e.* a rectified linear unit (ReLU)):

$$\mathbf{g} = \mathrm{ReLU}(\mathbf{W_g}\mathbf{h}^* + \mathbf{b_g}) \quad (6)$$

where $\mathbf{W_g} \in \mathbb{R}^{d_h}$ and $\mathbf{b_g}$ are a weight matrix and a bias.

We then apply a linear transformation that learns to map the sentence representation with d_h dimensions into the property embeddings with d_y dimensions as follows:

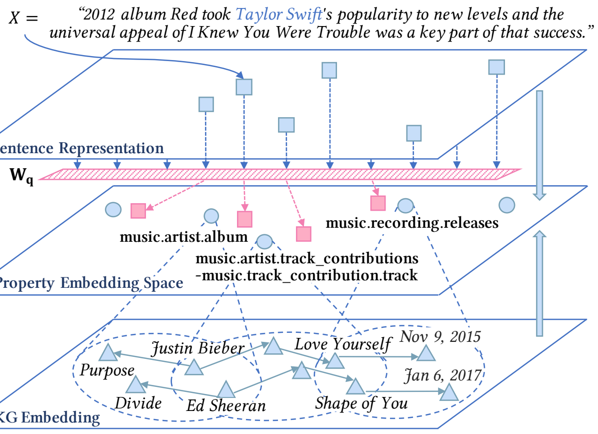
$$\mathbf{q} = \mathbf{W_q}\mathbf{g} + \mathbf{b_q} \quad (7)$$

where $\mathbf{W_q} \in \mathbb{R}^{d_y}$ and $\mathbf{b_q}$ are a weight transformation matrix and a bias, respectively.

To allow the model to learn the proper mapping, we utilize dot-product similarity to allow the model to learn to increase the similarity between the sentence representation and the embeddings of the represented properties. Given a property embedding matrix, $\mathbf{E_y} \in \mathbb{R}^{|Y'| \times d_y}$, where Y' represents a set of Y_{train} during training or Y_{test} during testing, and a sentence representation, $\mathbf{q}$, we compute the dot-product similarity vector $\mathbf{s} = \mathbf{E_y}\mathbf{q}$.

As illustrated in Figure 2, each sentence representation is transformed using the learned transformation matrix, $\mathbf{W_q}$, such that the sentence representation of the sentence, X, is close to the embeddings of the properties represented by the sentence (*i.e.* the properties **music.artist.album** and **music.artist.track_contributions –music.track_contribution.track**). As a result, the model produces a high similarity score for these properties. Moreover, since $\mathbf{W_q}$ learns to map the sentence representation of the same embedding space, $\mathbf{W_q}$ can be applied to the sentence representations of the properties, even though they are unseen during training.

3.2.4 Restricting properties by entity types. The next component is designed to increase the probability that the property represented in the sentence will be predicted by the model. To achieve this, we use the entity type of the target entity for the sentence. The entity type of each target entity indicates a group of properties into which a sentence should be classified. For example, given any sentence according to the target entity of entity type "music.artist", the represented properties should be among those of this entity type, such as **music.artist.genre** and **music.artist.album**, whereas the properties, such as **education.educational_ institution.school_type** and **education.educational_institution. campuses**, should never be predicted as being represented by any sentences of the target entity of entity type "music.artist". Thus, we



X = "2012 album Red took Taylor Swift's popularity to new levels and the universal appeal of I Knew You Were Trouble was a key part of that success."
Sentence Representation
W_q
music.artist.album
music.recording.releases
music.artist.track_contributions -music.track_contribution.track
Property Embedding Space
Justin Bieber
Love Yourself
Nov 9, 2015
Purpose
Jan 6, 2017
Divide
Ed Sheeran
Shape of You
KG Embedding

Figure 2: Mapping between sentence representation and property embeddings.

use the KB structure that allows recognition of the entity type of the entity, as well as the properties connected to the entity of the entity type. Given the target entity, e, for the sentence, X, there is an entity type of the entity, t_e, that corresponds to a set of properties, Y_{t_e}. Let $\mathbf{v}_{\mathbf{t_e}} \in \{0,1\}^{|Y'|}$ denote the entity type vector, where i-value is set to 1 if $y_i \in Y_{t_e}$, otherwise 0 and Y' denotes a set of Y_{train} during training or a set of Y_{test} during testing. The entity type vector $\mathbf{v}_{\mathbf{t_e}}$ is used to perform element-wise multiplication using the output from the previous component as $\mathbf{s}^* = \mathbf{s} \odot \mathbf{v}_{\mathbf{t_e}}$. In this way, the entity type vector suppresses prediction of irrelevant properties.

3.2.5 Sigmoid function. Finally, we apply a sigmoid function to compute probabilities over all properties $y \in Y'$ for a given sentence, X, as $P(y|X) = \sigma(\mathbf{s}^*)$.

3.3 Learning

For model training, we chose to apply cross-entropy as the objective function. Let y and $\hat{y}$ be the target and predicted probabilities over every property for each sentence, respectively. We define the objective function as follows:

$$L_{\mathrm{CLF}} = -\sum_i \sum_j y_{i,j} \log \hat{y}_{i,j} \quad (8)$$

where i denotes an index of sentences in the training data, and j denotes an index of classes.

We trained our model in an end-to-end fashion by adopting Adam [11] as our optimization algorithm.

3.4 Property Embedding

We have explained each of the model components and how the model works with the property embeddings to overcome the ZSL problem. In this section, we formally describe how the property embeddings are constructed. We begin this section by describing the characteristics of properties, given that they correspond to the predicate path. Later, we explain several techniques for constructing property embeddings based on a knowledge graph (KG) embedding using a well-known technique, TransE [1].

3.4.1 Property characteristics. The properties of the entities can be derived according to the structure of the KG which represents linkages between collections of triplets. Each triplet, (s, r, o), comprises a subject, predicate, and an object, where the predicate represents the relationship between the subject and object. The subject of the triplet can be connected by a predicate and be an object of another triplet, whereas the object of the triplet can be a subject of another triplet if it is connected with the other predicate. Accordingly, this typical graph characteristic allows exploration between subjects and objects through the predicate paths. Given a set of triplets, $\mathcal{K}$, a set of triplets with any predicate path is defined as $\mathcal{P} = \{(s, \mathbb{r}, o) | (s, r_1, x_1) \in \mathcal{K} \wedge (x_1, r_2, x_2) \in \mathcal{K} \wedge \cdots \wedge (x_n, r_n, o) \in \mathcal{K}\}$, where a predicate path $\mathbb{r} = (r_1, r_2, \cdots, r_n)$, x_i denotes an intermediate between the subject, s, and the object, o, for the predicate path $\mathbb{r}$, and n denotes the predicate path length. In this work, we defined the subject in $\mathcal{P}$ as the entity and the predicate path comprising a list of predicates in $\mathcal{P}$ as the property.

3.4.2 KG embedding by TransE. TransE is an energy-based model that learns low-dimensional embeddings of subjects, predicates, and objects based on triplets in $\mathcal{K}$. Given a triplet, (s, r, o), the predicate represented by a translation vector, $\mathbf{r}$, acts as a translation between the representation of the subject, $\mathbf{s}$, and the object, $\mathbf{o}$, such that $\mathbf{s} + \mathbf{r} \approx \mathbf{o}$. Such embeddings are learned by minimizing the margin-based ranking loss:

$$L_{\mathrm{KG}} = \sum_{(s,r,o) \in \mathcal{K}} \sum_{(s',r,o') \in \mathcal{K}'} [\gamma + d(\mathbf{s} + \mathbf{r}, \mathbf{o}) - d(\mathbf{s}' + \mathbf{r}, \mathbf{o}')]_+ \quad (9)$$

where $\mathcal{K}'$ denotes a set of corrupted triplets in which the subject or object is replaced by either a random subject or object, d denotes the dissimilarity measure, γ denotes the margin parameter, and $[x]_+$ denotes the positive part of x.

3.4.3 Representing property embedding. To obtain the embedding of each property, we use representations of subjects, predicates, and objects obtained by the KG embedding technique. We expect that the property embeddings will be close to each other in the embedding space if the properties are similar, and vice versa. Although the properties are constructed from the predicates, only representations of the predicates might not very well represent the similarity among seen and unseen properties. Thus, we chose to incorporate representations of subjects and objects instead of focusing only on the representations of predicates. We considered four techniques for representing properties, with each technique utilizing different components (Figure 3). Given a property, $y \in Y$, and a predicate path, $\mathbb{r}$, for the property y, we formulate an embedding, e_y, according to each of the techniques as follows:

- **SUM-R**: Summation of representations of predicates in the predicate path, $\sum_{r_i \in \mathbb{r}}^{n} \mathbf{r_i}$. For example, the embedding of the property **music.artist.track_contributions–music.track_contribution.track** is constructed by combining predicate representations *music.artist.track_contributions* and *music. track_contribution.track.*
- **CC-SO**: Concatenation of the average of the representations of subjects and objects, $[\bar{\mathbf{s}}, \bar{\mathbf{o}}]$. For example, we exploit the average representations of the subjects *Justin Bieber* and *Ed Sheeran*, as well as the average representations of the objects *Love Yourself* and *Shape of You*, in order to represent the property **music.artist.track_contributions–music.track_contribution.track** by the embedding.
- **CC-RO**: Concatenation of the representation of the first predicate in the predicate path and the average of the representations of objects, $[\mathbf{r}_1, \bar{\mathbf{o}}]$. For example, the embedding of the property **music.artist.track_contributions–music.track_contribution.track** is represented by the concatenation of the representation of the first predicate, *music.artist.track_contributions*, and the average representations of the objects, *Love Yourself* and *Shape of You.* We consider the first predicate in the predicate path as the most important predicate according to the embedding, because it is directly connected to the subject of the predicate path.
- **CC-SON**: Concatenation of the average of the representations of subjects, objects, and neighbor objects of neighbor predicates of the last predicate in the predicate path, $[\bar{\mathbf{s}}, \bar{\mathbf{o}}, \bar{\mathbf{n}}]$. With this technique, we can construct the embedding of the property **music.artist.track_contributions–music.track_contribution.track** by using the average representations of the subjects *Justin Bieber* and *Ed Sheeran*, the average representations of the objects *Love Yourself* and *Shape of You*, and also the average representation of the neighbor object *Vocals.* This technique is similar to the second technique; however, we also used the neighbor objects of the predicate path that could help differentiate one property from another sharing some of the subjects and objects, although they may not be similar to each other.

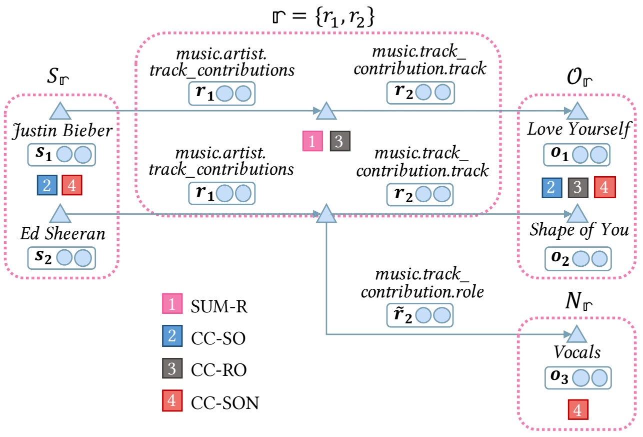


Figure 3: Example components used in each of the four techniques to represent property embeddings using KG embedding. This assumes that the property is derived from a predicate path, $\mathbb{r}$, which comprising $n(= 2)$ predicates.

The components for these techniques are computed as $\bar{\mathbf{s}} = \frac{\sum_{s \in S_{\mathbb{r}}} \mathbf{s}}{|S_{\mathbb{r}}|}$, $\bar{\mathbf{o}} = \frac{\sum_{o \in O_{\mathbb{r}}} \mathbf{o}}{|O_{\mathbb{r}}|}$, and $\bar{\mathbf{n}} = \frac{\sum_{o \in N_{\mathbb{r}}} \mathbf{o}}{|N_{\mathbb{r}}|}$ if $n > 1$, otherwise $\mathbf{0}$, where $S_{\mathbb{r}} = \{s \mid (s, \mathbb{r}, o) \in \mathcal{P}\}$, $O_{\mathbb{r}} = \{o \mid (s, \mathbb{r}, o) \in \mathcal{P}\}$, $N_{\mathbb{r}} = \{o \mid (s, (r_1, r_2, \cdots, r_{n-1}, \tilde{r}_n), o) \in \mathcal{P}\}$, $\tilde{r}_n$ represents a neighbor predicate of the predicate r_n, and n denotes the predicate path length.

4 EXPERIMENTS

Our goal is to identify a set of properties represented in sentences given a target entity. The proposed model is capable of classifying the sentences according to their representation of specific entity properties and handling the ZSL problem. The experiments provided evidence addressing the following research questions:

- **RQ1**: How effectively can our proposed model handle the ZSL problem relative to baseline methods?
- **RQ2**: Which technique(s) is the best for representing properties with embeddings in our case?
- **RQ3**: Under what situation(s) can the model work effectively for handling the ZSL problem?

To this end, we first explain the datasets used for our experiments, followed by a description of the evaluation metrics. We then show our experimental settings before presenting and discussing the experimental results.

4.1 Datasets

We used two datasets to evaluate our model. The first is a widely used dataset (NYT10) originally released by Riedel *et al.* [22] and generated by aligning triplets in Freebase with a New York Times (NYT) corpus. The dataset comprises **54** properties. However, since we expect our method to exhibit high-coverage identification of the properties, we preferred a larger number of properties. Additionally, the small number of properties in the existing dataset might not allow effective evaluation of the model efficacy for handling the ZSL problem. Therefore, we developed our own dataset (WEB19) comprising **271** properties. In the following subsection, we describe the process of generating our dataset in detail.

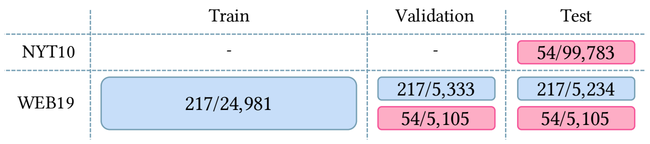
4.1.1 Dataset - WEB19. To prepare this dataset, we first selected a set of properties based on the existing predicate paths in the FB15k dataset. The FB15k dataset was published by Bordes *et al.* [1] and was used to generate the KG embedding for our classification model. Details of the FB15k dataset are reported in Section 4.3.2. We selected the predicate path as the property in situations where it associates with more than 100 triplets in Freebase[1] and is among the top seven to fifteen predicate paths by the entity type sorted from the highest to lowest number of triplets in Freebase.

After selecting the set of predicate paths as properties, we obtained up to 500 subject and object pairs for each property for sentence extraction via a Web search engine (*i.e.* Bing). For each pair, we issued a search query containing the subject and object names to Bing Web Search API[2] and extracted every sentence that included the subject and object names within the top 20 search results to represent each property. However, the set of extracted sentences of each property included some sentences that did not properly represent the property, even though they contained the subject and object names of the property. Additionally, this approach allows us to achieve only one property represented by each sentence, whereas in reality, one sentence can represent more than one property simultaneously.

To improve our dataset, we employed crowdsourcing workers on Amazon Mechanical Turk to classify a set of extracted sentences. Since the number of extracted sentences was large, we randomly selected up to 500 extracted sentences per property. For each sentence, we allocated three workers and asked them to judge

[1]https://developers.google.com/freebase

[2]https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api



	Train	Validation	Test
NYT10	-	-	🟥 54/99,783
WEB19	🟦 217/24,981	🟦 217/5,333	🟦 217/5,234
		🟥 54/5,105	🟥 54/5,105

Figure 4: Number of properties and sentences in the train, validation, and test sets for the NYT10 and WEB19 datasets. The blue and pink boxes indicate sets of sentences for seen and unseen properties, respectively.

whether the properties in a group were represented in the sentence, given the subject (*i.e.* the entity) appearing in the sentence along with the description of the properties. For each sentence, we prepared a group of properties including "NA" which indicates that none of the properties was represented in the given sentence.

We computed a Free-Marginal Multirater Kappa [20] to measure the agreement in worker responses and found a moderate result at 0.602. We regarded the properties that received a majority of votes among workers to be those represented in the sentence, and we assigned "NA" to the remaining sentences with properties not receiving a majority of the votes.

For both the NYT10 and WEB19 datasets, we excluded sentences associating with "NA". For the WEB19 dataset, we also excluded properties associated with the number of sentences less than 30 and used only the remaining sentences for the experiments.

Figure 4 illustrates the separation of the two datasets along with the numbers of properties and sentences in each set. We regarded every property in the NYT10 dataset to be the unseen properties in the test set, while for our WEB19 dataset, we first randomly selected properties to represent seen and unseen properties, followed by separation of the sentences associating with the seen properties into training, validation, and test sets. For the sentences associating with the unseen properties, we separated these sentences into validation and test sets and used the validation set to tune the hyperparameters.

4.2 Evaluation Metrics

We evaluated variations of our proposed model and the baseline methods using the three test sets (*i.e.* one test set for seen properties and the other two for unseen properties) from the NYT10 and WEB19 datasets, and reported the results based on two groups of evaluation metrics (*i.e.* sample-based and label-based).

4.2.1 Sample-based. The results were evaluated based on each sample sentence using **accuracy**, **precision**, **recall**, **F-measure**, and **hit-at-k**. The first four metrics are defined as follows:

$$\mathrm{A}_{\text{sample}} = \frac{|Y_s \cap \hat{Y}_s|}{|Y_s \cup \hat{Y}_s|} \quad (10) \qquad \mathrm{P}_{\text{sample}} = \frac{|Y_s \cap \hat{Y}_s|}{|\hat{Y}_s|} \quad (11)$$

$$\mathrm{R}_{\text{sample}} = \frac{|Y_s \cap \hat{Y}_s|}{|Y_s|} \quad (12) \qquad \mathrm{F}_{\text{sample}} = \frac{2|Y_s \cap \hat{Y}_s|}{|Y_s| + |\hat{Y}_s|} \quad (13)$$

where Y_s represents a set of ground truth properties for a sentence, s, and $\hat{Y}_s$ is a set of properties that are predicted with probabilities higher than 0.5 for a sentence, s.

For hit-at-k, this determines how well the model predicts at least one represented property within the top-k items of the ranked list of predicted properties. This metric is computed for each sentence, s, as follows: hit@k = 1 if $\hat{Y}_s^k \cap Y_s \neq \emptyset$; otherwise 0, where $\hat{Y}_s^k$ is a set of properties within top-k for a sentence, s.

4.2.2 Label-based. For this group of metrics, the results were evaluated according to each property label over all sample sentences. To compute the label-based metric, we used **F-measure**, which can be computed as follows: $F_{\text{label}} = \frac{2|S_l \cap \hat{S}_l|}{|S_l| + |\hat{S}_l|}$, where S_l is a set of sentences with property label l, and $\hat{S}_l$ is a set of sentences with property label l predicted with probabilities higher than 0.5.

4.3 Experimental Settings

4.3.1 Word embedding. We trained a Skip-gram model [15] on the English Wikipedia articles[3]. In the experiments, we used word embeddings with 100, 200, and 300 dimensions.

4.3.2 KG embedding. To obtain the KG embedding, we relied on the original dataset (FB15k) published by Bordes *et al.* [1] consisting of three sets of triplets from Freebase. Each triplet comprises a predicate path, with the maximum length at two. However, we did not directly utilize this dataset, because it does not include any predicate paths with objects as non-entities (*e.g.* numerical values).

In this study, we extended the dataset to include additional predicate paths. We first combined the triplets from the three sets and split the predicate paths, $\mathbb{r}$, with lengths higher than 1 into the individual predicates and completed the missing subject or object of each predicate, r, with the subject or object in Freebase based on two conditions: (1) the subject of the triplet should be the subject of the predicate, r_1, and the predicate of the triplet is the predicate r_1; and (2) the object of the triplet should appear as the subject of the other triplet, for which the predicate is the predicate r_2, and the object is the object of the predicate, r_2. As a result, the object of this triplet was added as the object of the predicate, r_1, whereas it was added as the subject of the predicate r_2. We then obtained the additional predicate paths with lengths equal to 1 by finding triplets with subjects the same as those recently added. These triplets were then added to the dataset.

For simplicity, we combined the predicate paths (all having a length of 1) in the NYT10 dataset in order to train KG embedding and use the trained model to evaluate both the NYT10 and WEB19 datasets. We prepared the Freebase triplets for the NYT10 predicate paths and added them to the dataset. We then separated the triplets into three sets keeping only those where the subjects and objects appeared in at least three triplets as the subjects or objects and the predicates appeared in at least five triplets. We then separated the remaining triplets by allowing all unique subjects, predicates, and objects to reside in the training set. We ultimately obtained 1,186,193, 330,388, and 330,388 triplets in the training, validation, and test sets, respectively, with the number of unique subjects and objects at 209,319, and the number of unique predicates at 2,206. For the experiments, we used the embeddings of subjects, predicates, and objects with 50 and 100 dimensions. Moreover, we followed [1] by setting all parameters for training the KG embedding to be the same as the optimal settings for the FB15k dataset.

[3]https://dumps.wikimedia.org/enwiki

4.3.3 Comparison of the proposed models and baseline methods. We compared variations of our proposed model with random baselines as well as the existing methods which applied the neural network-based models for the relation extraction task. The baseline methods are listed as follows:

- **Rand and Rand-T**: Randomly selected n properties for sentences based on all properties and a group of properties for the entity type of the target entity. The number of properties n was randomly selected using the probability distribution of the number of properties across all sentences in the WEB19 dataset.
- **RNN**: RNN model [26].
- **CNN**: CNN model [35].
- **BRNN**: Bidirectional RNN model [36].
- **LSTM**: LSTM model.
- **BLSTM**: Bidirectional LSTM model.

For each variation of our proposed model and each baseline method, except for Rand and Rand-T, we used the size of the hidden layer as 50 and trained them using a batch size of 32 sentences.

For comparison, we presented the result of each variation of our proposed model and each baseline method, except for Rand and Rand-T, by the best performance achieved among the combination of hyperparameters (*i.e.* word embedding and KG embedding dimensions) based on the validation set of the WEB19 dataset.

4.4 Results

4.4.1 **RQ1** - *How effectively can our proposed model handle the ZSL problem relative to baseline methods?* Table 1 shows the performance of baseline methods and variations of our proposed model in terms of mean accuracy, precision, recall, F-measure, and hit@k (*i.e.* k=1) using sample sentences of properties for both the NYT10 and WEB19 datasets. To answer RQ1, we focused on method performance at identifying *unseen* properties. The results indicated that the proposed model #7 utilizing both entity types and property embeddings constructed by the CC-SO technique outperformed every baseline method, with the highest mean accuracy at 0.441, precision at 0.441, recall at 0.683, F-measure at 0.511, while the highest mean hit@1 was achieved by the proposed model #9 that utilized the entity types and property embeddings using the CC-SON technique on the NYT10 dataset. For the WEB19 dataset, proposed model #6 that utilized the entity types and property embeddings constructed by the SUM-R technique achieved the best performance according to all evaluation metrics (A_{sample} = 0.395, P_{sample} = 0.414, R_{sample} = 0.450, F_{sample} = 0.418, and hit@1 = 0.420). The performance in identifying unseen properties achieved by our best model #6 on the WEB19 dataset was comparable to that achieved for identifying seen properties with mean accuracy at 0.322, precision at 0.402, recall at 0.377, F-measure at 0.366, and hit@1 at 0.416.

Although most baseline methods cannot handle the ZSL problem, our proposed models using the property embeddings (*i.e.* models #2-9) were able to handle this problem according to their higher performances especially for models #6-9 on both datasets. A similar trend was observed with proposed model #1, which did not utilize the property embeddings. Although the model #1 achieved

Table 1: Results for each method in identifying properties for the target entity in terms of mean accuracy, precision, recall, F-measure, and hit@1 using sample sentences from the NYT10 and WEB19 datasets. We considered variations of our proposed model for comparison with baseline methods. Each of the proposed models are represented as ① for the use of the entity type of the target entity, ② for the use of the property embeddings, and ③ for the technique constructing the property embeddings.

	Method				NYT10					WEB19									
					Unseen properties					Unseen properties					Seen properties				
					A_{sample}	P_{sample}	R_{sample}	F_{sample}	hit@1	A_{sample}	P_{sample}	R_{sample}	F_{sample}	hit@1	A_{sample}	P_{sample}	R_{sample}	F_{sample}	hit@1
Baseline	Rand				0.002	0.004	0.006	0.004	0.003	0.005	0.005	0.006	0.005	0.008	0.005	0.006	0.007	0.006	0.007
	Rand-T				0.268	0.323	0.402	0.344	0.336	0.162	0.175	0.178	0.172	0.179	0.129	0.181	0.173	0.159	0.181
	RNN				0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.012	0.031	0.041	0.032	0.035	0.212
	CNN				0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.019	0.145	0.188	0.162	0.164	0.270
	LSTM				0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.026	0.137	0.170	0.167	0.157	0.294
	BRNN				0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.021	0.147	0.183	0.178	0.168	0.339
	BLSTM				0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.016	0.161	0.198	0.197	0.185	0.315
	#ID	①	②	③															
Proposed Model	#1	✓	-	-	0.000	0.000	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.022	0.331	**0.423**	0.373	0.374	**0.518**
	#2	-	✓	SUM-R	0.015	0.015	0.413	0.030	0.002	0.028	0.028	0.336	0.051	0.017	0.154	0.194	0.193	0.179	0.308
	#3	-	✓	CC-SO	0.021	0.021	0.467	0.041	0.001	0.136	0.142	0.234	0.164	0.229	0.169	0.206	0.227	0.199	0.306
	#4	-	✓	CC-RO	0.013	0.013	0.307	0.025	0.001	0.056	0.058	0.230	0.090	0.012	0.162	0.197	0.224	0.192	0.302
	#5	-	✓	CC-SON	0.022	0.022	0.343	0.040	0.004	0.101	0.107	0.170	0.121	0.234	0.171	0.207	0.231	0.201	0.317
	#6	✓	✓	SUM-R	0.328	0.328	0.498	0.369	0.325	**0.395**	**0.414**	**0.450**	**0.418**	**0.420**	0.322	0.402	0.377	0.366	0.416
	#7	✓	✓	CC-SO	**0.441**	**0.441**	**0.683**	**0.511**	0.433	0.283	0.298	0.337	0.306	0.297	0.321	0.401	0.390	0.370	0.422
	#8	✓	✓	CC-RO	0.400	0.400	0.677	0.466	0.398	0.354	0.369	0.429	0.380	0.372	0.323	0.402	0.387	0.370	0.421
	#9	✓	✓	CC-SON	0.435	0.435	0.656	0.502	**0.471**	0.282	0.295	0.343	0.306	0.297	**0.333**	0.416	**0.392**	**0.380**	0.435

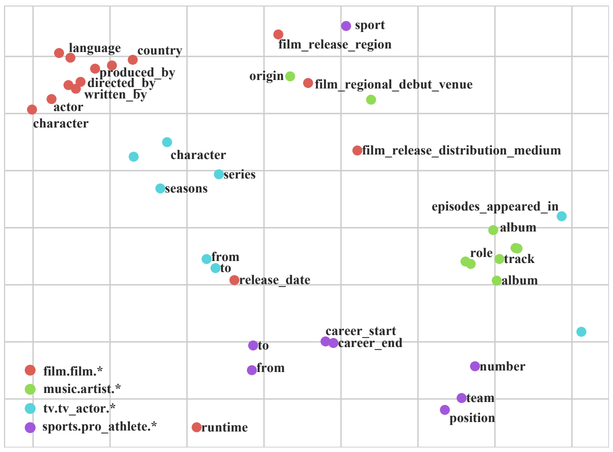


Figure 5: The embeddings of properties constructed by SUM-R technique of the four entity types on the WEB19 dataset comprising a large number of properties on the embedding space.

relatively high performance in identifying seen properties comparing with models #6-9 on the WEB19 dataset, this model was incapable of identifying unseen properties. This can be interpreted that property embeddings are crucial for allowing the models to identify unseen properties.

Moreover, comparison of the performances of models #2-5 with models #6-9 revealed that models #2-5, which only utilized property embeddings by different techniques achieved lower performance relative to models #6-9 that used both the entity type and property embeddings. This confirmed the importance of the entity type of target entity, given that they help filter out the properties that should not be predicted as represented in sentences regarding the target entity. Therefore, in order to effectively handle the ZSL situation, the models do not only require property embeddings, but also the entity type of the target entity.

4.4.2 **RQ2** - *Which technique(s) is the best for representing properties with embeddings in our case?* The experimental results on Table 1 for the NYT10 dataset showed that many unseen properties can be accurately identified using proposed model #7, as it achieved the highest mean accuracy at 0.441, precision at 0.441, recall at 0.683, and F-measure at 0.511. Accordingly, we can infer that CC-RO was the best for representing unseen properties on the NYT10 dataset, because it was the technique used by proposed model #8. Meanwhile, SUM-R was the best technique for representing unseen properties on the WEB19 dataset, given that proposed model #6 utilizing this technique outperformed other proposed models utilizing other techniques, with the highest mean accuracy at 0.395, precision at 0.414, recall at 0.450, F-measure at 0.418, and hit@1 at 0.420. However, the different technique (*i.e.* CC-SON) used by proposed model #9 achieved the best performance in terms of mean accuracy (0.333), recall (0.392), and F-measure (0.380).

Since the best performances were achieved by the different techniques on the different datasets, we considered another important factor which is the number of seen properties for each entity type during model training. We found that the average number of seen properties for the entity type on the NYT10 dataset was 0.919, whereas the average number of seen properties for the entity type on the WEB19 dataset was much larger at 7.692. With a large number of seen properties for the entity type, the properties needed to be *precisely* represented in the embedding space using the representations of predicates so that the unseen properties can be accurately identified by the model. Thus, for datasets with a sufficiently large number of seen properties of each entity type, the techniques using the representations of predicates (*i.e.* SUM-R and CC-RO) could better handle the ZSL problem in identifying unseen properties. On the other hand, the NYT10 dataset that includes a small number of seen properties for each entity type might require the properties to be *loosely* represented in the embedding space to

Table 2: Best and worst relative performances in identifying unseen properties on the WEB19 dataset by the best proposed model #6 as compared with the best baseline method (Rand-T) in terms of label-based F-measure (F_{label}).

		Unseen properties	F_{label}	# of seen properties
Best	1	**organization.organization.headquarters–location.mailing_address.street_address_2**	4.078	13
	2	**education.educational_institution.subsidiary_or_constituent_schools**	3.526	10
	3	**film.film.release_date_s–film.film_regional_release_date.release_date**	3.46	12
	4	**tv.tv_program.regular_personal_appearances–tv.tv_regular_personal_appearance.person**	2.757	12
	5	**organization.organization.leadership–organization.leadership.title**	2.505	13
Worst	1	**education.field_of_study.academics_in_this_field**	0	6
	2	**location.statistical_region.gni_in_ppp_dollars–measurement_unit.dated_money_value.amount**	0	5
	3	**government.governmental_body.sessions**	0	10
	4	**base.schemastaging.organization_extra.phone_number–base.schemastaging.phone_sandbox.country_code**	0	5
	5	**baseball.baseball_player.batting_stats–baseball.batting_statistics.batting_average**	0	8

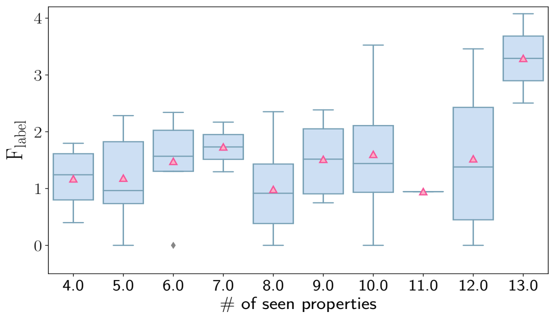


Figure 6: Relative performance in identifying unseen properties on the WEB19 dataset of the best proposed model #6 that utilized the entity type and the property embeddings constructed by SUM-R technique as compared with the best baseline method (Rand-T) in terms of label-based F-measure (F_{label}) relative to the number of seen properties of the same entity type.

achieve the high performance in identifying unseen properties. For the properties to be loosely represented in the embedding space, the representations of subjects and objects are useful because they allow the unseen properties to potentially be similar to the seen properties of the other entity types. Based on this reason, the CC-SO and CC-SON techniques utilized in the models #7 and #9 could achieve high performances on the NYT10 dataset.

To illustrate how the properties were embedded by t-SNE [14], Figure 5 depicted the embeddings of properties of the four major entity types using the SUM-R technique, which is the best technique to identify unseen properties on the WEB19 dataset. We found that similar properties in the same entity type are close to each other such as the properties **produced_by**, **directed_by**, and **written_by** of the entity type "film.film" as well as the properties **seasons** and **series** of the entity type "tv.tv_actor". Moreover, properties of the different entity types can be closely embedded in the embedding space if they are expressed by similar syntactic patterns. For example, the properties **music.artist.origin** and **film.film.release_date–film.film_regional_release_date.film_regional_debut_venue**. However, some properties belonging to the same cluster are too close to each other such as the clusters representing properties of the entity types "film.film" and "music.artist". With such circumstance, the model might not well distinguish a certain property from another.

4.4.3 **RQ3** - *Under what situation(s) can the model work effectively for handling the ZSL problem?* Figure 6 illustrates the relative performance in identifying unseen properties on the WEB19 dataset using the best proposed model #6, as described in Table 1, relative to the best baseline method (Rand-T) in terms of label-based F-measure (F_{label}) according to the number of seen properties of the same entity type. We found that the higher maximum F_{label} was achieved if a larger number of seen properties could be used during model training: the proposed model showed better performances for unseen properties when the number of seen properties of the same entity type was between 9 and 13.

The best and worst unseen properties on the WEB19 dataset in terms of model #6 are shown in Table 2. These examples support the hypothesis that higher performances can be achieved if there are many seen properties of the same entity type: the five unseen properties that achieved the best F-measure scores were associated with a larger number of seen properties in the same entity type, while the five unseen properties that achieved the worst F-measure scores were associated with a smaller number of seen properties.

Thus, a larger number of seen properties for each entity type may be critical for our proposed model to effectively handle the ZSL problem.

5 CONCLUSION

We proposed the neural network-based model for multi-label sentence classification that classifies sentences as representing properties given a target entity. We emphasized the concern over the ZSL problem when the training sentences of certain properties are unavailable. To cope with this problem, we utilized property embeddings constructed by different techniques based on components of the KG embedding. The model also utilized an entity type of the target entity to narrow the potential properties represented in the sentences. Experimental results showed that utilizing both the entity type and property embeddings improves the performance in identifying both seen and unseen properties.

Acknowledgements This work was supported by JSPS KAKENHI Grant Numbers 18H03244, 18H03243, 16H02906, 17H06099, and 18H04093, and JST PRESTO Grant Number JPMJPR1853, Japan. We thank Tokyo ACM SIGIR Chapter for the discussion at POWIR.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[2] Horatiu Bota, Ke Zhou, and Joemon M Jose. 2016. Playing your cards right: The effect of entity cards on search behaviour and workload. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*. ACM, 131–140.

[3] Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. 576–583.

[4] Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*. Association for Computational Linguistics, 724–731.

[5] Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 423.

[6] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*. 2121–2129.

[7] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 267–274.

[8] Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 427–434.

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[10] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 541–550.

[11] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[12] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 2014. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 3 (2014), 453–465.

[13] Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, et al. 2015. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*. 4247–4255.

[14] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[16] Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 777–782.

[17] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics, 1003–1011.

[18] Raymond J Mooney and Razvan C Bunescu. 2006. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*. 171–178.

[19] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. 2010. Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th international conference on World wide web*. ACM, 771–780.

[20] Justus J Randolph. 2005. Free-Marginal Multirater Kappa (multirater K [free]): An Alternative to Fleiss' Fixed-Marginal Multirater Kappa. *Online submission* (2005).

[21] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. 2016. Learning deep representations of fine-grained visual descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 49–58.

[22] Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 148–163.

[23] Alan Ritter, Luke Zettlemoyer, Oren Etzioni, et al. 2013. Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics* 1 (2013), 367–378.

[24] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[25] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*. 935–943.

[26] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. Association for Computational Linguistics, 1201–1211.

[27] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. Association for Computational Linguistics, 455–465.

[28] Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, 721–729.

[29] Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. 2016. Latent embeddings for zero-shot classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 69–77.

[30] Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vol. 2. 665–670.

[31] Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. 1785–1794.

[32] Xiaoxin Yin and Sarthak Shah. 2010. Building taxonomy of web search intents for name entity queries. In *Proceedings of the 19th international conference on World wide web*. ACM, 1001–1010.

[33] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research* 3, Feb (2003), 1083–1106.

[34] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1753–1762.

[35] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 2335–2344.

[36] Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006* (2015).

[37] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vol. 2. 207–212.