Supporting Entity-oriented Search with Fine-grained Information in Knowledge Graphs

WIRADEE IMRATTANATRAI

Department of Social Informatics
Graduate School of Informatics
Kyoto University

Unfortunate events though potentially a source of anger and despair have equal potential to be a source of spiritual growth. Whether or not this is the outcome depends on your response.

– Dalai Lama XIV

Abstract

Users usually search for information related to entities such as celebrities, politicians, and companies via search engines. Many modern search engines have been trying to improve user's search experience in terms of entity-oriented search by introducing search functionalities that directly present relevant entity-related information based on a user's information need on the search engine result page. With such functionalities, the search users could complete their search tasks regarding their entities of interest easily with less efforts since they do not need to explore as many documents in organic search results. Normally, the entity-oriented search functionalities are empowered by a large-scale Knowledge Graph (KG) as the KG contains structured information of entities gathered from various sources. Nonetheless, although there exists a handful of entity-related information within the KG, the search engines sometimes fail to provide the direct information of entities due to the complexity and ambiguity of the search queries. Since some search queries are lengthy as they contain different specific terms or phrases regarding the entities or the entity types such as a query "top selling movies julia roberts during 80's" which contains several specific phrases "top selling" and "during 80's" with respects to the entity type "Movies by Julia Roberts", without the potential to properly interpret these phrases, the search engines could end up providing irrelevant results to the users. In contrast, some search queries can be ambiguous and under-specified as the user might issue a plain and simple query "julia roberts" that does not indicate any specific terms representing aspects of the entity "Julia Roberts". With this type of queries, it is difficult for the search engines to prepare the relevant entity-related information unless they can infer the actual user's information need. Thus, in this thesis, we aim to support the users in searching for the entity-related information by leveraging fine-grained entity information in terms of different data representation and different complexity level of the information within the KG. We propose three research problems where each of them could help mitigating the difficulty in searching for entity-related information in heterogeneous ways. Each of the research proposals is listed as follows:

- First, we focus on *the ranking of entities for queries with modifiers*. For example, given a search query "ancient temples in kyoto", the goal is to return a ranked list of entities temples (*e.g.* "Kennin-ji", "Tenryu-ji", and "Kinkaku-ji") according to the modifiers "ancient" and "kyoto". To achieve this, we utilize qualitative properties that indicate categorical information of entities (*e.g.* **city**) and quantitative properties that indicate numerical information of entities (*e.g.* **founded date**) in the KG to filter and rank the entities according to the modifier terms in the queries. Thus, we could increase the coverage of the search queries that expect a ranked list of related entities to be returned so that the users can directly obtain the relevant

entities for a more number of queries.

- Second, we focus on *the identification of entity properties from text with zero-shot learning*. For example, given 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 from text is considered as a fundamental task that is essential and can be applied to several entity-oriented search functionalities. For instance, the identified entity properties can be used to construct a more relevant entity card, *i.e.* one of the successful entity-oriented search functionalities. In addition, we incorporate a zero-shot learning (ZSL) that could help to solve a task despite not having enough training sentences for some properties, especially those properties that are quantitative properties and are represented by complex information within the KG. With the ZSL, the higher number of properties could be identified as we could integrate many types of properties for entities.
- Third, we focus on *the explanation of relationship between entities*. For example, given a set of entities including "The Lion King", "Tim Rice", "Elton John" and "Disney", we aim to provide the explanation "Tim Rice wrote the songs with Elton John for Disney animated film The Lion King" as it expresses how the given entities are related. For this proposal, we attempt to make use of the complex information of multiple entities in the KG since those information could be useful for providing descriptive explanation of entity relationship. Providing the explanation of the relationship of entities to the users could reduce the difficulty of the users in understanding the relatedness between entities since there are multiple entity-oriented search functionalities involving multiple entities. One good example of those functionalities is entity recommendation where it recommends a set of entities relevant to the given entity. Without any explanation of how the suggested entities are related to the given entity, the users might hesitate to put any attentions to any suggested entities. Hence, good relationship explanation of entities would help relieving this complication between the search users and the search engines.

Acknowledgement

This work could have remained incomplete without the participation of the following persons who have contributed and supported in this work.

I would like to express my sincere gratitude to my supervisor, Professor Masatoshi Yoshikawa, who gave helpful advices, guidance and the opportunities for having very useful discussions.

I would like to express my deepest thanks my advisors during my whole doctoral course, Professor Keishi Tajima, Professor Shinsuke Mori and Professor Sadao Kurohashi, for their advice and helpful comments during every discussion.

Besides my supervisor and advisors, I would like to thank Emeritus Professor Katsumi Tanaka for always giving me support and encouragement since the beginning of my doctoral course even after his retirement.

I am also very much thankful to respected faculty and fellow lab members for their beneficial comments and suggestions during lab meetings.

I would like to express my very great appreciation to Associate Professor Makoto P. Kato, who has always been giving me his kind support, encouraging guidance and valuable comments. He has always been sharing his precious knowledge and thought upon this work. His dedication and commitment are indeed very important that drive this work until its completion.

I would like to acknowledge with gratitude the love and support from my partner, Mr. Suppanut Pothirattanachaikul, who has always been there with me during not only the good, but also those tough time. He has always encouraged me to do things that I do not even believe myself I could accomplish.

Last, but definitely not least, I would like to thank my family for their love and support throughout my study.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Entity-oriented Search

Searching for information related to *entities* such as celebrities, movies, and travel attractions has become prevalent among search users. As suggested in previous studies, about 71% of search queries contain entity names (*e.g.* a search query "julia roberts movies" which contains the name of the entity "Julia Roberts"), and at least 20-30% of them are simply entity names (*e.g.* a search query "julia roberts" which simply represents the entity "Julia Roberts") [1–3]. To enable more efficient

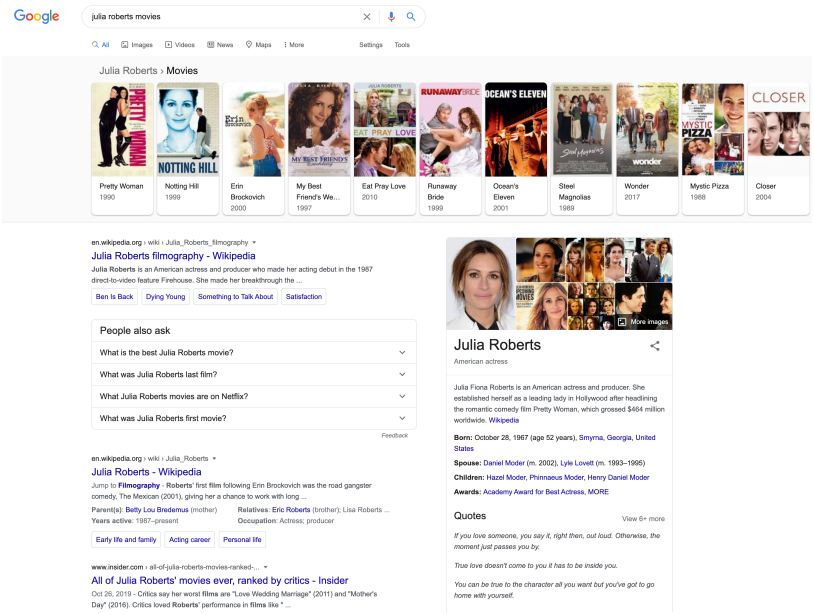


Figure 1.1: The actual SERP by Google for the search query "julia roberts movies" which includes the organic search results (*i.e.* a list of retrieved documents) along with the entity-oriented search functionalities (*i.e.* entity carousel at the top, and entity card on the right of the SERP).

access to the entity-related information, most modern search engines such as Google[1] , Yahoo![2] and Bing[3] have introduced *entity-oriented search functionalities* that directly present entity-related information in conjunction with organic search results on a search engine result page (SERP). The examples of the well-known functionalities include *entity carousel* which presents a list of entities relevant to a user's search query and *entity card* which presents a short entity's summary, a set of facts and a list of suggested entities according to a focused entity in a user's search query. The actual results of the two functionalities can be seen in Figure 1.1. With the support from these functionalities, it affects the search users to accomplish their search tasks regarding the entities with less time-consuming since they do not have to explore many documents in a long list of retrieved documents.

1: https://www.google.com/
2: https://www.yahoo.co.jp/
3: https://www.bing.com/

1.2 Knowledge Graph

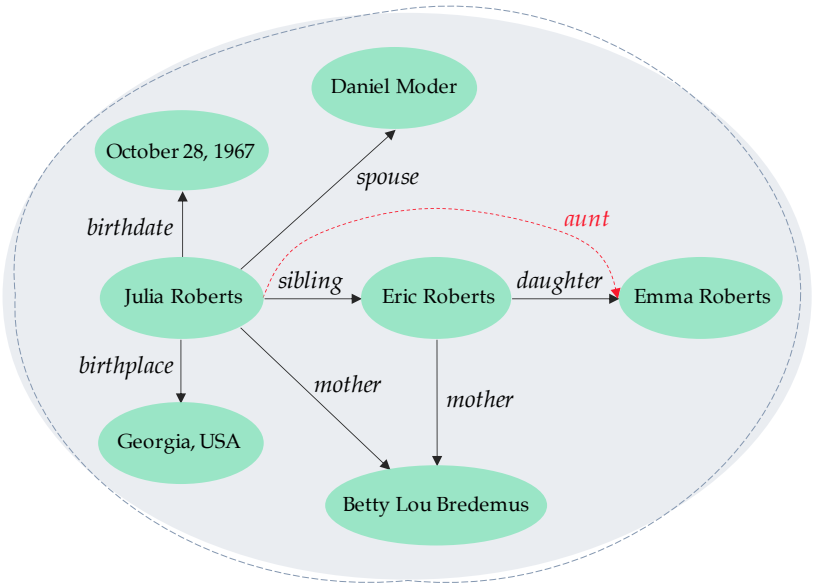


Figure 1.2: The example subgraph of the KG which represents the information related to the entity "Julia Roberts".

To support users with the entity-oriented search functionalities, a Knowledge Graph (KG) is a great source for the search engines to obtain the information about the entities. The KG comprises a large set of entities and their information connected by predicates such as a predicate *spouse*, *sibling* and *birthdate* as shown in Figure 1.2. The KG is constructed based on a Knowledge Base (KB) consisting of a collection of triplets. Each triplet comprises a subject, predicate, and an object, where the predicate represents the relationship between the subject and object. For example, a triplet ("Julia Roberts", *spouse*, "Daniel Moder") which indicates the spouse relationship between the entity "Julia Roberts" as the subject and the entity "Daniel Moder" as the object, and a triplet ("Julia Roberts", *birthdate*, [October 28, 1967]) which indicates the birth date of the entity "Julia Roberts". In addition, any connected predicates (*i.e.* a predicate path) between any subjects and objects of the paths in the KG can be referred to as **properties** such as the property **aunt** between the entity "Julia Roberts" as the subject and the entity "Emma Roberts" as the object of the path ("Julia Roberts", *sibling*, "Eric Roberts", *daughter*, "Emma Roberts") which represents the predicate path **sibling-daughter** consisting of a sequence of predicates *sibling* and *daughter*.

1.3 Current Limitations

With a handful of available entity-related information in the KG, the search engines could be able to cover users' information needs regarding the entities to some extent. However, the search engines sometimes fail to provide relevant entity-related information regarding the user's information need due to the complexity and ambiguity of the search queries.

In terms of the complexity, users sometimes issue long queries with different specific terms or phrases regarding the entities or the entity types such as a query "top selling movies julia roberts during 80's"

which contains several specific phrases "top selling" and "during 80's" with respects to the entity type "Movies by Julia Roberts". Without the potential to properly interpret these phrases, the search engines could end up providing irrelevant results to the users.

On the other hand, some search queries can be ambiguous and under-specified as the user might issue a plain and simple query "julia roberts" that does not indicate any specific terms representing aspects of the entity "Julia Roberts" based on user's information needs, though there are so many aspects regarding to this entity such as personal life, career, and education. While each aspect of the entity corresponds to a different set of relevant entity-related information, it is difficult for the search engines to prepare the relevant entity-related information without knowing the actual user's information need.

1.4 Our Research Proposals

Accordingly, this thesis aims at alleviating current limitations of the entity-oriented search functionalities by exploiting the fine-grained information within the KG in terms of different data representation and different complexity level of the information to its full potential. Our study is separated into three research topics where each of them focuses on improving a different element of the current entity-oriented search functionalities. Figure 1.3 illustrates the overview of the three research proposals with their inputs and expected outputs. We explain each research proposal and its advantages for users on the entity-oriented search as follows:

- First, we focus on *the ranking of entities for queries with modifiers*. For example, given a search query "ancient temples in kyoto", the goal is to return a ranked list of entities temples (*e.g.* "Kennin-ji", "Tenryu-ji", and "Kinkaku-ji") according to the modifiers "ancient" and "kyoto". To achieve this, we utilize qualitative properties that indicate categorical information of entities (*e.g.* **city**) and quantitative properties that indicate numerical information of entities (*e.g.* **founded date**) in the KG to filter and rank the entities according to the modifier terms in the queries. Thus, we could increase the coverage of the search queries that expect a ranked list of related entities to be returned so that the users can directly obtain the relevant entities for a more number of queries.
- Second, we focus on *the identification of entity properties from text with zero-shot learning*. For example, given 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 from text is considered as a fundamental task that is essential and can be applied to several entity-oriented search functionalities. For instance, the identified entity properties can be used to construct a more relevant entity card, *i.e.* one of the successful entity-oriented search functionalities. In addition, we incorporate a zero-shot learning (ZSL) that could help to solve a task despite not having enough training sentences for some properties, especially those

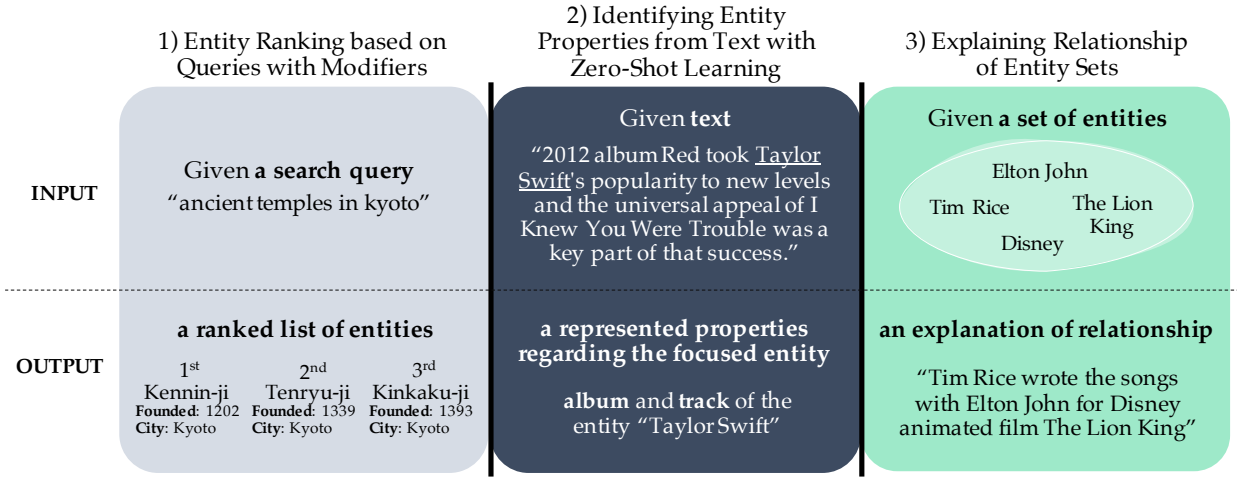


Figure 1.3: The input and the expected output of each of the three proposed research problems including: 1) Entity Ranking based on Queries with Modifiers, 2) Identifying Entity Properties from Text with Zero-Shot Learning, and 3) Explaining Relationship of Entity Sets.

properties that are quantitative properties and are represented by complex information within the KG. With the ZSL, the higher number of properties could be identified as we could integrate many types of properties for entities.

- Third, we focus on *the explanation of relationship between entities*. For example, given a set of entities including "The Lion King", "Tim Rice", "Elton John" and "Disney", we aim to provide the explanation "Tim Rice wrote the songs with Elton John for Disney animated film The Lion King" as it expresses how the given entities are related. For this proposal, we attempt to make use of the complex information of multiple entities in the KG since those information could be useful for providing descriptive explanation of entity relationship. Providing the explanation of the relationship of entities to the users could reduce the difficulty of the users in understanding the relatedness between entities since there are multiple entity-oriented search functionalities involving multiple entities. One good example of those functionalities is entity recommendation where it recommends a set of entities relevant to the given entity. Without any explanation of how the suggested entities are related to the given entity, the users might hesitate to put any attentions to any suggested entities. Hence, good relationship explanation of entities would help relieving this complication between the search users and the search engines.

1.5 Comparison of Our Proposals with Existing Literature

We discuss and explain the main differences between each of our research proposals and the existing studies (*i.e.* the ranking of entities, the identification of entity properties from text, and the explanation of relationship between entities) in this section. The comparison are listed as follows:

- The problem of returning a ranked list of entities in the existing works [4–9] focused on the search queries including the entity

type name and terms that describe the specific characteristic of the entities such as the term "adventure" describing the genre of the entities movies. These terms are referred to as *modifiers* and are very important for finding relevant entities, however, most existing works only focused on the modifiers corresponding to the qualitative properties (*e.g.* the modifier "adventure" corresponding to the qualitative property **genre**) although there exists the modifiers corresponding to the quantitative properties (*e.g.* the modifier "ancient" corresponding to the quantitative property **founded date**) that usually appear as a part of the search queries such as the query "ancient temples in kyoto". The modifiers that correspond to the quantitative properties are as important as the modifiers corresponding to the qualitative properties and require additional interpretation as well as more in-depth analysis in order to find relevant entities. Therefore, in this research, we focus on returning a ranked list of entities based on search queries by considering both modifiers corresponding to qualitative properties and modifiers corresponding to quantitative properties.

- The problem of identifying entity properties from text is similar to relation extraction task, where one of the most promising approaches involves supervised learning through distant supervision [10–13]. With the distant supervision, it allows the training sentences to be efficiently obtained by using triplets of the predicate path representing the property in the existing KB as the approach does not require excessive human annotations. However, for a certain property, it is likely that a few or none of the sentences could be obtained if the predicate path representing the property is associated with a small number of triplets. Moreover, for some entity properties such as the quantitative properties that represent numerical data and those that represent the implicit relationship (*i.e.* a predicate path with more than two predicates as properties), it could be even more difficult to obtain sufficient amount of training sentences. A lack of training sentences for some properties results in the inability to learn the text pattern in order to identify those unseen properties (*i.e.* the properties without training sentences). This problem is well-known as the zero-shot learning (ZSL) problem. Being unable to identify certain properties could result in low coverage of properties, and more importantly, result in the users not acquiring important information concerning the entity. Thus, in this study, we focus on increasing the coverage of the entity properties to be identified in text. To enable the high of coverage of the entity properties, we focus on solving the ZSL problem when identifying entity properties.
- The problem of explaining a relationship between entities using text has previously been addressed in some existing works [14–16]. However, their approaches were specifically designed for explaining a pair of entities based on an explicit relationship which can simply be represented by a single hop path (*i.e.* a predicate) in the KG. When considering more than two entities, the existing approaches might not effectively be applied as the relationship of a set of entities is represented by multi-hop paths where each path includes different combinations of entities and predicates. Thus, we focus on generating the explanation of relationship between

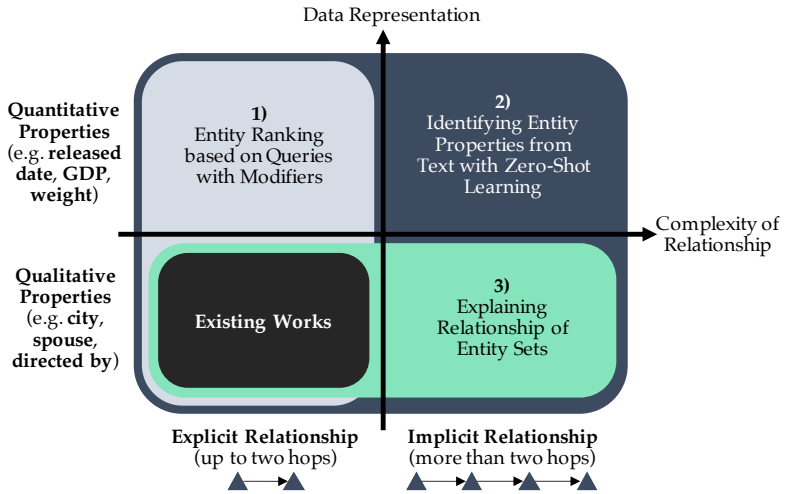


Figure 1.4: The overall thesis framework by contrasting the three proposed research problems (represented in three different color boxes including grey, dark blue and green, respectively) and the existing works (represented in black box) in terms of the data representation (*i.e.* vertical axis) and the complexity of the relationship (*i.e.* horizontal axis) within the KG.

a set of entities. In this case, we do not only emphasize on the explicit relationship, but also the implicit relationship between entities which is represented by a multi-hop path. The implicit relationship is as important as the explicit relationship and is very useful to be integrated in entity-oriented search functionalities. At the same time, exploiting the implicit relationship in entity-oriented search functionalities is more challenging and required more sophisticated interpretation than the explicit relationship.

To this end, we attempt to solve the different limitations of the existing studies based on each research proposal. As shown in Figure 1.4 which illustrates the overall comparison between our proposals and the existing works in terms of data representation and complexity of relationship of the entity-related information in the KG, for the first proposed research problem (*i.e.* Entity Ranking based on Queries with Modifiers), we focus on returning a ranked list of entities based on search queries by considering both modifiers corresponding to qualitative properties and modifiers corresponding to quantitative properties, while most existing studies considered on only modifiers corresponding to qualitative properties. For the second proposed research problem (*i.e.* Identifying Entity Properties from Text with Zero-Shot Learning), we focus on solving the ZSL problem when identifying entity properties from text so that we could increase the coverage of the properties to be identified. In this case, we could consider multiple types of properties including both quantitative and qualitative properties, and more importantly, we could make use of properties representing more complex relationship. Lastly, for the third research problem (*i.e.* Explaining Relationship of Entity Sets), we focus on generating the explanation of relationship between a set of entities by fully utilizing the KG as we aim to also leverage multi-hop paths in the KG which represent more complicate relationship of entities. Overall, our three researches attempt to incorporate higher level of detailed of entity-related information as well as higher complexity level of the relationship of entities in the KG.

1.6 Thesis Structure

The remainder of this thesis is structured as follows:

- **Chapter 2**: We survey and discuss the existing works that are related to each of our research proposals
- **Chapter 3**: We present the propose methods of finding a ranked list of entities for a given query by leveraging different types of modifiers in the query through identifying corresponding properties. While most major search engines provide the entity search functionality that returns a list of entities based on users' queries, entities are neither presented for a wide variety of search queries, nor in the order that users expect. To enhance the effectiveness of entity search, we propose two entity ranking methods including a Web-based entity ranking and a property-based entity ranking. Thus, we propose a novel property identification method that identifies a set of relevant properties based on a Support Vector Machine (SVM) using our proposed seven criteria that are effective for different types of modifiers. The experimental results showed that our proposed property identification method could predict more relevant properties than using each of the criteria separately. Moreover, we achieved the best performance for returning a ranked list of relevant entities when using the combination of the Web-based and property-based entity ranking methods.
- **Chapter 4**: We propose a method for identifying a set of entity properties from text with ZSL. 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.
- **Chapter 5**: We propose a learning framework for generating a textual relationship explanation for a set of entities. While a problem of explaining the relationship of entities has previously been addressed, the existing approaches were specifically designed for a pair of entities concerning an explicit relationship within the KG. When considering multiple entities, their relationship tends to be complex and could be explained using multi-hop paths. Thus, we explore a new approach in which we fully utilize the information and structure of the KG through a reinforcement learning with an encoder-decoder network to generate a relationship explanation for a set of entities. By using our newly constructed

dataset, experiments revealed that our proposed learning framework achieved better performance and could generate a more descriptive relationship explanation than baselines.

- **Chapter 6**: We summarize our three research proposals, address the technical and social contributions, as well as discuss the future directions.

Related Work 2

In this chapter, we survey the existing works in terms of three main topics including: 1) Entity Ranking, 2) Property Identification from Text, and 3) Explanation of Relationship between Entities.

2.1 Entity Ranking

This section introduces related works on finding a ranked list of entities and identifying relevant properties.

One of the most related works is *keyword search* over structured and semi-structured data [17–20]. Elbassuoni *et al.* proposed a retrieval model for the keyword search over RDF graphs. This retrieval model retrieves a set of subgraphs in which their elements (or entities) match with the keyword queries, and ranks them by considering the predicates (or properties) based on statistical language-models [17].

In addition to the keyword search over RDF, systems such as DISCOVER [18], DBXplorer [19] and BANKS [20] were developed to support the keyword search over relational databases. The idea is to return tuples containing given keywords using joining operations between relations in databases. For DISCOVER [18] and DBXplorer [19], the returned tuples are ranked by the number of joins required, whereas BANKS [20] ranks them based on the edge weight computation and prestige based ranking (*e.g.* PageRank [21]).

Similar to the keyword search problem, *question answering* over RDF graphs is also considered as one of the related works as they find answers (or entities) based on questions (or queries) with the use of the relevant properties for modifiers in the question [22, 23]. Zou *et al.* proposed methods to find relevant predicates or predicate paths (or properties) using supporting entity pairs for the relation phases (or modifiers) [22], while Unger *et al.* proposed methods that use the pattern library extracted by the BOA framework [24] for finding relevant properties [23].

Furthermore, some related works are concerned with *entity ranking*, which has been addressed in some tracks in INEX and TREC [25–28]. The INEX entity ranking track is comprised of two tasks: entity ranking and entity list completion tasks. The entity ranking task expects systems to return relevant Wikipedia articles (or entities) in response to a given query where there is assumption that all entities have the corresponding pages in Wikipedia. Kaptein *et al.* proposed methods to rank Wikipedia entities by estimating relevant Wikipedia categories for a given query, and rank entities based on the query likelihood model with estimated categories [4]. Demartini *et al.* expanded queries for the entity ranking task in many ways such as using hierarchical relationship in the ontology and synonyms, and extracting named entities from the queries [5]. Balog *et al.* proposed a probabilistic framework, which can model not only

keyword queries, but also the other inputs such as categories in which relevant entities should belong to and examples of relevant entities [6].

The entity track in TREC 2010 introduced another task called *related entity finding*. This task is related to the entity ranking tracks in INEX, but the main input is an entity name or a homepage of an entity and output is a list of homepages of entities. Bron *et al.* proposed methods for this task which reduce problems in ranking by using four components including co-occurrence, type filtering, context modeling, and homepage finding [7].

More study related to the problem of finding properties corresponding to modifiers was authored by Zhang *et al.* [29]. In this work, relevant properties (of *facets*) were estimated for a given query in order to provide better snippets for structured documents. While a machine learning approach was employed for finding relevant properties, most of the features were textual similarity that often used for learning to rank for Web search (*e.g.* TF-IDF or BM25).

The problem of returning a ranked list of entities in this research is different comparing to other related research works. The differences can be listed as 1) The query types in which we are interested in are different and 2) our returned ranked lists of entities are also associated with the ranking orders of the modifiers that correspond to the quantitative properties. Since our focused queries contain the modifiers that further indicate more specific characteristics of entities, these modifiers are not usually included in the element names within structured and semi-structured data, or even the Wikipedia articles. As a result, we cannot directly retrieve entities based on the similarity between the query and those predefined names. In this case, we not only use textual similarity, but also utilize Web search results using the original search queries, and the relevant properties of the modifiers. Moreover, as we interested in both modifiers corresponding to the qualitative and quantitative properties, the entities need to be filtered based on the qualitative properties as well as ordered based on the quantitative properties. According to our survey, other related research works focus on modifiers corresponding to qualitative properties, which are required in filtering entities. On the other hand, the modifiers corresponding to quantitative properties are used to order the entities. This type of modifiers requires additional effort since we need to determine the ranking order of the entities according to the modifiers (*i.e.* ascending and descending order) and then rank the entities in the right order.

Moreover, the existing approaches for finding relevant properties such as the textual similarity are not suitable for modifiers corresponding to quantitative properties, for which we propose methods based on the co-occurrence of terms on the Web and property value distributions of entities.

2.2 Property Identification from Text

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 [30–34]. 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.* [10–13]). For distant supervision, sentences are automatically aligned with the corresponding properties based on triplets in an existing KB. Mintz *et al.* [10] 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 [11–13, 35] 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.* [36] 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.* [37] presented a passage retrieval model that provides relevance feedback for filling in missing triplets. Similar to [37], Min *et al.* [38] and Ritter *et al.* [39] addressed the problem of an incomplete KB. Min *et al.* [38] utilized a semi-supervised approach that models true bag-level labels as the extension to [13], whereas Ritter *et al.* [39] introduced a new latent-variable approach that models missing triplets as the extension to [12].

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 sentences as representing certain properties [40–45]; 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 some 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 both seen and unseen classes modeled using external information, such as class attributes [46], word vectors [47–

49] and text description [50, 51]. 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 [52]).

2.3 Explanation of Relationship between Entities

We surveyed the existing studies regarding the problem of explaining a relationship of entities using textual explanations as well as other types of explanations such as paths and subgraphs.

Voskarides *et al.* [14] casted the problem as a sentence ranking problem as they tried to rank candidate sentences for an entity pair using a learning-to-rank approach according to each predicate in the KG. Similarly, Huang *et al.* [15] utilized clickthrough data to rank sentences via a pairwise ranking model. Meanwhile, Voskarides *et al.* [16] relied on generating explanations using predefined templates of a path or subgraph of a pair of entities. However, for a sentence ranking approach, a sufficient number of candidate sentences for an entity set could be hardly obtained unless the entities are well-known and their relationship is likely to be described in many available sources. In addition, the template-based approach requires a large number of predefined templates to cover a diverse combination of predicates as a path. Thus, we focused on the explanation generation task where we do not need to rely on both candidate sentences and templates.

Another line of works attempted to generate text by utilizing the given KG [53–56]. These works used and interpreted the content of the whole graph as relevant and useful for generating text. Thus, their graphs are relatively small. To generate a descriptive relationship explanation for our problem, it is important to discover an informative path that contains crucial information about entities in the KG. To find an informative path, previous studies emphasized on graph structure properties for measuring the informativeness of the candidate paths or subgraphs [57–65]. Besides, Kasneci *et al.* [59] proposed to also consider semantic measures such as a co-occurrence of entities and relationships on a graph. Another line of researches relied on predicting a one-hop path between a pair of entities by modeling embeddings of entities and predicates [52, 66–68]. To discover more complex relationships that are represented by multi-hop paths, Lao *et al.* [69] applied a random walk with restart over the KG using their previously proposed Path-Ranking Algorithm (PRA) [70] based on a given entity pair and a relationship. Alternately, Neelakantan *et al.* [71] and Toutanova *et al.* [72] proposed neural models to model representations of paths extracted by the PRA, then computed the scores over all representations based on a given relationship. A more recent work adopted a multi-hop reasoning over the KG by exploiting a reinforcement learning to find paths between a pair of entities [73] such that the approach does not require any previously extracted paths.

Entity Ranking based on Queries with Modifiers

3

3.1 Introduction

Entity search has been introduced as one of the recent emerging trends in most major search engines such as Google, Yahoo!, and Bing. This functionality supports search users in exploring entities more directly than finding entities from a collection of Web documents. The entity search is increasingly important as previous studies reported that about 71% of queries contain entity names, and at least 20-30% of them are simply entity names [2, 3]. The large amount of these kind of queries suggests that users often look for entity-related information while they are searching.

A major challenge in the entity search is to deal with complex queries since queries have been becoming longer for expressing much more specific information needs [74, 75]. Some long queries form sentences and contain types of entities to be retrieved as well as *modifiers*. For example, "*highest profitable* companies in japan" and "*upcoming american mystery* movies". Modifiers in these examples are "highest profitable", "upcoming", "american" and "mystery". Modifiers are usually used to further narrow down entities to be retrieved. However, the current entity search functionality does not filter or sort entities by considering many of the modifiers, especially modifiers that correspond to properties whose values are numerical, *i.e.* quantitative properties (*e.g.* the modifier "highest profitable" corresponding to the property **operatingIncome** of company entities), as it requires additional interpretation as well as more in-depth analysis in order to find relevant entities based on these modifiers. Moreover, the existing methods only focus on modifiers corresponding to properties whose values are categorical, *i.e.* qualitative properties (*e.g.* the modifier "mystery" corresponding to the property **genre** of movie entities), even though modifiers corresponding to quantitative properties are as useful as other modifiers for narrowing down the list of entities. Hence, it is necessary to interpret both types of modifiers to fully satisfy user needs in the entity search.

To overcome these limitations, we propose two entity ranking methods that return a ranked list of entities for queries with modifiers.

The first proposed entity ranking method, *Web-based entity ranking*, directly finds highly relevant entities from Web search results returned in response to the query as a whole, and propagates the estimated relevance to the other entities. The second proposed entity ranking method, *property-based entity ranking*, ranks entities based on relevant properties of each modifier in the query. For identifying relevant properties, we propose a method based on a machine learning approach using our proposed seven criteria.

In the experiments, we used 50 queries to evaluate the effectiveness of our proposed methods. The results showed that our property identifi-

cation method could identify more relevant properties when all criteria were combined using the SVM, and achieved the best performance for returning a ranked list of entities when using the combination of the Web-based and property-based entity ranking methods.

The contributions of this study are:

1. We introduced two entity ranking methods for queries including modifiers: the Web-based entity ranking and property-based entity ranking.
2. We proposed a property identification method to be used in the property-based entity ranking based on our proposed seven criteria that are suitable for different types of modifiers.
3. We conducted experiments to evaluate the performance of the proposed methods and demonstrated the effectiveness of our methods over the existing methods.

3.2 Methodology

In this section, we propose two entity ranking methods: Web-based entity ranking and property-based entity ranking methods. We also propose a property identification method to identify relevant properties for the property-based entity ranking method.

3.2.1 Problem Definition

Before going through each proposed method, we formally define our problem in this subsection. Our problem is to return a ranked list of entities for a given query $q \in Q$. We are specifically interested in queries Q that contain an entity type name and modifiers. A modifier is an attributive term that describes a typical feature of entities. Modifiers can be categorized into two types in our research: ones corresponding to qualitative properties (*e.g.* "comedy", "new york" and "outsourcing"), and ones corresponding to quantitative properties (*e.g.* "oldest", "large" and "highest profitable").

- **Definition 1 (Qualitative properties)**: properties in which their property values are *nominal* or *ordinal* that cannot be measured, *e.g.* properties **genre**, **location**, and **award**.
- **Definition 2 (Quantitative properties)**: properties in which their property values are *interval* or *ratio* that can be measured, *e.g.* properties **established**, **weight**, and **operatingIncome**.

To return a list of entities for a given query, we utilize RDF data from a knowledge base that represents the relationship across entities E, entity types T, and properties P. Entities of the entity type $t \in T$ are denoted by E_t, entity types to which entity $e \in E$ belongs are denoted by T_e, and a set of properties of entity $e \in E$ are denoted by P_e.

In the following subsections, we describe each of our proposed methods for returning a ranked list of entities.

3.2.2 Web-Based Entity Ranking

The first proposed method directly finds highly relevant entities for a given query based on a Web search engine, and propagates the estimated relevance to the other entities based on the entity similarity. Since we assume that a given query contains entity type names, entities should be relevant if the names of the entity types they belong to are similar to the query. Thus, we use the BM25 to measure their similarity. Additionally, relevant entities may frequently appear in Web search results returned in response to the query as some Web pages list up entities and describe them with the same words in the query. We further explore relevant entities by using Co-HITS [76] for propagating the estimated relevance to the other similar entities.

3.2.2.1 BM25 Score between Query and Entity Type

We measure the similarity between query $q \in Q$ and entity type $t \in T$ using the BM25, which is denoted by $\mathrm{sim}(q, t)$. We regard entity types whose $\mathrm{sim}(q, t)$ is θ% (0.8 in our experiments) of the maximum score or larger as a set of relevant entity types as $T_q = \{ t \mid \mathrm{sim}(q, t) \geq \theta \max_{t' \in T}(\mathrm{sim}(q, t')) \}$. Thus, candidates of relevant entities E_q should belong to these entity types: *i.e.* $E_q = \bigcup_{t \in T_q} E_t$.

3.2.2.2 Co-HITS Algorithm using BM25 Score and Document Frequency

The BM25-based approach can only measure the relevance of entities in terms of their entity types, but not take into account modifiers in the query. Since there are Web pages introducing some entities with description including the same words in the query, one can know a *subset* of relevant entities by issuing the given query into a Web search engine and finding entity names within the search results. For example, there can be some Web pages introducing the highest profitable companies and can be retrieved through a Web search engine with query "highest profitable companies". As we emphasized, however, entities found by this approach can be only a subset of relevant entities. Thus, we propose a method to propagate the BM25 score and frequency on the Web search results with the Co-HITS algorithm.

We first issue the original search query q to a Web search engine and collect top D search results R_q (D was set to 20 in our experiments). For each candidate entity $e \in E_q$, we count the number of search results that include the entity name in their content as the document frequency. The document frequency is defined as $\mathrm{df}(e, R_q) = |\{ r \mid r \in R_q \wedge e \in E_r \}|$ where E_r is a set of entities whose name is included in the content of search result r.

Given the BM25 scores of entity types and document frequency of entities, we utilize the Co-HITS algorithm [76] to combine and propagate these two scores based on two assumptions: 1) entity types are likely to be relevant if they contain relevant entities, and 2) entities are likely to be relevant if they belong to relevant entity types. As entities that have a high document frequency are likely to be relevant to query q, we can infer

that entity types including such entities are also relevant. In a similar way, we can further infer that entities included in relevant entity types are also relevant. The Co-HITS algorithm can be used to make these inferences and enables us to find more relevant entities and entity types that would be missed when using methods described above. The Co-HITS algorithm can be described as following equations:

$$c(q,t) = (1-\lambda_T)\,\mathrm{sim}(q,t) + \lambda_T \sum_{e\in E_q} w_{et}\,c(q,e), \tag{3.1}$$

$$c(q,e) = (1-\lambda_E)\,\mathrm{df}(e,R_q) + \lambda_E \sum_{t\in T_q} w_{te}\,c(q,t), \tag{3.2}$$

where $c(q,t)$ and $c(q,e)$ are the Co-HITS scores of entity types and entities, w_{et} is the edge weight from entity e to entity type t, w_{te} is the edge weight from t to e, and λ_T and λ_E are the parameters that control the effect of the BM25 score and document frequency on the Co-HITS score (1.00 and 0.25 were turned out to be the optimal, respectively, in our preliminary experiments and were used in our experiments). The edge weights indicate is-a relationship between the entity types and entities and are defined as $w_{et} = 1/|T_e|$ and $w_{te} = 1/|E_t|$ if entity e belongs to entity type t; otherwise, 0.

Computing $c(q,e)$ for each entity, we can finally obtain a list of ranked entities E_q^* in descending order of $c(q,e)$. Although this approach is simple and effective as can be seen in our experiments, we can use a complementary method, the property-based entity ranking method, with this Web-based entity ranking method and further improve the retrieval performance.

3.2.3 Property-Based Entity Ranking

This proposed method involves with three consecutive tasks: 1) modifier detection (detecting terms in a given search query to be modifiers), 2) property identification (identifying a set of relevant properties for modifiers), and 3) entity ranking by relevant properties (ranking entities using a set of relevant properties).

3.2.3.1 Modifier Detection

We firstly need to detect the modifiers in the query. To this end, we assume that the modifiers are terms that are not used for indicating the relevant entity types, *i.e.* ones that do not contribute to the BM25 scores of the relevant entity types. Specifically, for each term w in query q, we sum up all the BM25 scores of the entity types in T_q that contain term w in their entity type names, as $s(w) = \sum_{t\in T_q \wedge w\in n(t)} \mathrm{sim}(q,t)$, where $n(t)$ represents a set of terms in the name of entity type t. Then, we select term w as a modifier only if the sum of the BM25 scores for term w is relatively low. The set of modifiers in query q is denoted by M_q and obtained as $M_q = \left\{ w \;\middle|\; s(w) \le \phi \sum_{w'\in q} s(w') \right\}$, where ϕ is a parameter that we set to 0.2 in our experiments. The left term in the inequality is the sum of BM25 scores for term w, while the right term is that for all the terms in the query.

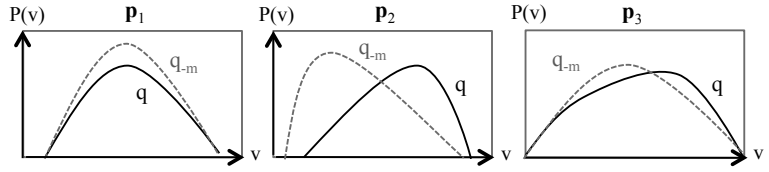


Figure 3.1: Distributions of q and q_{-m} for each property (p_j) where x-axis (v) represents the property value of p_j, and y-axis ($P(v)$) represents probability density of property value v. Here, largest distribution difference can be seen with property p_2, which is possibly relevant to modifier m.

For each modifier $m \in M_q$, we attempt to find a corresponding property from properties $P_{E_q^*}$ ($= \bigcup_{e \in E_q^*} P_e$) that can appropriately describe the modifier.

3.2.3.2 Property Identification

The detected modifiers are used to identify relevant properties based on the seven criteria. The details of seven criteria are discussed in the following subsections.

Criterion 1: Frequency of Property Values The first criterion is to count the frequency of property values for each property p that contain modifier m. Some modifiers indicate a certain type of entities (*e.g.* "comedy" is a property value of relevant property **genre** and "new york" is included in the property values of the relevant property **location**) and often specifies the property values of relevant properties. The frequency of property values including modifier m is computed as follows:

$$\text{freq}(q, m, p) = |\{ e \mid e \in E_q^* \wedge m \in v(e, p) \}|, \tag{3.3}$$

where E_q^* is a set of candidate entities for query q and $v(e, p)$ represents a set of property values of property p for entity e.

Criterion 2: Co-occurrence of Modifiers and Properties on the Web The second criterion is to measure the co-occurrence of modifiers and property names on the Web. When people describe the characteristic of entities by modifiers (*e.g.* "old", "large", and "highest profitable"), they are likely to show their property values as evidence (*e.g.* "foundation date", "area size", and "income", respectively). Thus, a high co-occurrence between a modifier and a property name indicates that the modifier specifies a certain property value of the property. We compute how frequently modifier m co-occurs with property p by using Web search results from the original query q as follows:

$$\text{co}(m, p) = \frac{|\, S_{mp} \,|}{|\, D_p \,|}, \tag{3.4}$$

where S_{mp} is the set of (adjacent) sentences in the Web pages containing modifier m and property p, and D_p is the set of web pages containing property p.

Criterion 3: Difference between Property Value Distributions The third criterion is to measure the difference in property value distributions of entities in search results based on two queries 1) an original query (*e.g.* "large companion dog breeds") 2) a query without the modifier

corresponding to qualitative properties (*e.g.* "large dog breeds" without the modifier "companion") or a query with the antonym of the modifier corresponding to quantitative properties (*e.g.* "small companion dog breeds" with the antonym of the modifier "large"). We expect that Web search engines will return search results containing different set of entities when using different queries. These different sets of entities would have different property values which relevant to terms in the query. Based on this assumption, we can measure the change in property value distributions of each property and estimate that properties that cause big changes are relevant to modifiers. Figure 3.1 shows some examples of pairs of property value distributions. We estimate that property p_2 is relevant to modifier m since the distributions for query q and q_{-m} are significantly different.

The property value distribution of property p for query q can be estimated as follows:

$$P_p^q(v) = \frac{|\{\, e \mid e \in E_{R_q} \wedge v \in v(e,p) \,\}|}{\sum_{v' \in V_p} |\{\, e \mid e \in E_{R_q} \wedge v' \in v(e,p) \,\}|}, \tag{3.5}$$

where v is a property value ($v \in V_p$; V_p is the set of all possible property values for property p), $v(e,p)$ is a set of property values of property p for entity e, and E_{R_q} is a set of entities included in the search results for query q (formally defined as $E_{R_q} = \bigcup_{r \in R_q} E_r$).

Letting q_{-m} be query q with the antonym of modifier m or without modifier m (*i.e.* $q - \{m\}$ if q is represented as a set of terms), the difference between property value distributions for queries q and q_{-m} is measured by Kullback-Leibler divergence as follows:

$$\mathrm{diff}(q,m,p) = P_p^q P_p^{q-m} = \sum_{v \in V_p} \log P_p^q(v) \frac{P_p^q(v)}{P_p^{q-m}(v)}. \tag{3.6}$$

Criterion 4: Similarity between Properties and Modifiers The fourth criterion is to compute the similarity between the name of each property p and modifier m. As some modifiers (*e.g.* the modifier "highest profitable") indicate the property names or the synonyms of the properties (*e.g.* the property **profit**, **income** and **revenue**), these properties have the same or similar meaning with the modifier and are likely to be relevant to the modifier m. To compute the similarity, we first constructed a distributional representation of words by word2vec [77] using the English version of Wikipedia articles (3,831,719 articles in total). Then, we compute the similarity between each property p and modifier m using the word vector representation by the word2vec model as follows:

$$\mathrm{sim}(m,p) = \frac{1}{|n(m)|} \sum_{w \in n(m)} \mathbf{v}_w^T \mathbf{v}_p, \tag{3.7}$$

where $n(m)$ represents a set of terms of modifier m, and $\mathbf{v}_w$ and $\mathbf{v}_p$ are word vectors of term w in $n(m)$ and property p, respectively.

Criterion 5: Co-occurrence of Major Property Value Types and Modifiers on the Web The fifth criterion is to measure the co-occurrence between major property value types and modifiers on the Web. Modifiers

are not always described on the Web pages using the property names since the major property value types (*e.g.* ***pounds***) can sometimes be adequate for interpreting the modifiers without explicitly indicating the property names (*e.g.* the property **weight**). For instance, within the top Web search results by the query "large dog breeds", the Web contents contain sentences describing the characteristic of large dogs by the modifier "large" and the major property value type ***pounds*** of the property **weight**, *i.e. "Large dogs typically tip the scales at 55 to 85* ***pounds****"* and *"Some groups define* ***large*** *dog breeds as those heavier than 100* ***pounds****"*. The property is likely to be relevant to the modifier if the major property value type of this property frequently co-occurs with the modifier. The major property value type of each property (y_p^*) can be obtained as follows:

$$y_p^* = _{y \in Y_{V_p}} | \{ e \mid e \in E_{R_q} \wedge y(e,p,v) = y \} |, \tag{3.8}$$

where Y_{V_p} is the set of all possible property value types for the property values of property p, E_{R_q} is a set of entities included in the search results for query q and $y(e,p,v)$ is the property value type of the property value v of the property p for the entity e.

After getting the major property value type of each property p, the co-occurrence of y_p^* with the modifier m can be computed as follows:

$$\text{co}(m, y_p^*) = \frac{| S_{m y_p^*} |}{| D_{y_p^*} |}, \tag{3.9}$$

where $S_{m y_p^*}$ is a set of (adjacent) sentences in the Web pages containing modifier m and y_p^*, and $D_{y_p^*}$ is the set of web pages containing y_p^*.

Criterion 6: Difference between Co-occurrence of Property Values and Entity Names on the Web The sixth criterion is to measure the difference of the co-occurrence of property values and entity names on the Web for the original query q and the query without modifier q_{-m}. As the Web search results returned for each search query q contain the Web contents that are expected to be relevant to query q, the entities that are described in the content are likely to be followed by the property values of the properties which help supplementing the content to be more relevant to query q. For example, when issuing the query "old temples in japan", the temples could frequently appear in the contents of Web search results more with the established date than with the name of the founder. However, when issuing the query "temples in japan" without the modifier "old", we could see that the temples are less likely to appear with the established date in the Web content. Therefore, we compute the difference between co-occurrence of the name of entity e with the property values of each property p on the Web by using the original query q and the query without modifier q_{-m}. We compute the co-occurrence of the name of entity e with the property values of each property p as follows:

$$C_p^q = \sum_{r \in R_q} \sum_{s_i \in r}^{|r|} | \{ e \mid e \in E_{R_q} \wedge n(e) \subset s_i \wedge v(e,p) \subset s_i \} |, \tag{3.10}$$

where R_q is a set of Web search results by query q, s_i is a sentence in the content of the Web search result r, E_{R_q} is a set of entities included in the

search results for query q, $n(e)$ is a set of terms in the name of entity e, and $v(e, p)$ is a set of property values of property p for entity e.

Then, we can compute the difference between the co-occurrence of the entity name with the property values of each property for the original query and the query without modifier on the Web as follows:

$$diff_{co}(q, m, p) = \frac{C_p^q}{C_p^{q-m}}, \tag{3.11}$$

If property p has high difference between the co-occurrence of the entity name with the property values of each property for the original query and the query without modifier on the Web, this property p is likely to be a relevant property for modifier m as the property values of property p are frequently used to explain the entities on the Web using the original query q.

Criterion 7: Frequency of Properties The seventh criterion is to compute the frequency of properties. This criterion is similar to the first criterion, but we count the number of entities whose property name appears as the modifier instead of the property value since it is possible that the modifier could directly indicate the property name as was mentioned earlier in the fourth criterion. We also used this seventh criterion because the fourth criterion measures only the similarity between the modifier and property names. In most cases, if the similarity between the modifier and property name is high, there is also high possibility that this property could be relevant to the modifier. However, we need to consider the number of entities that are associated with the property as well because if the number of entities is very low, we might not effectively rank the entities by using this property. The frequency of properties including modifier m is computed as follows:

$$\text{freq}_\text{p}(q, m, p) = |\{ e \mid e \in E_q^* \wedge m \in p_e \wedge p \in p_e \}|, \tag{3.12}$$

where E_q^* is a set of candidate entities for a query q and p_e represents a set of properties associated with entity e.

Support Vector Machine using Seven Criteria for Identifying Relevant Properties Our proposed property identification method utilizes all seven criteria described previously as features in the SVM classification. Since each criterion is applicable for a different type of modifiers, all criteria need to be employed collectively for boosting performance in finding relevant properties for the modifier. We used a Radial Basis Function (RBF) as a kernel function to the SVM as it is commonly used to take into account the interaction of features in the SVM.

We selected the property with the highest predicted score by the SVM, $z(q, m, p)$, for each modifier $m \in M_q$ and finally obtained a set of properties as $P_q^* = \{ {}_{p \in P_{E_q^*}} z(q, m, p) \mid m \in M_q \}$.

Ranking order	**Property value score** $s(e, m, p)$
Descending order	$\frac{v - \min_{e \in E_q^*} v(e,p)}{\max_{e \in E_q^*} v(e,p) - \min_{e \in E_q^*} v(e,p)}$
Ascending order	$\frac{v - \max_{e \in E_q^*} v(e,p)}{\max_{e \in E_q^*} v(e,p) - \min_{e \in E_q^*} v(e,p)}$

Table 3.1: Property value scoring for the quantitative properties depending on the ranking order of the modifier.

3.2.3.3 Entity Ranking by Relevant Properties

We rank entities using the relevant properties by firstly identifying the property type of the relevant property as different types of properties are used to differently rank the entities. Then, we assign the property value score for each property value of the property for entities depending on the ranking order correlated to each modifier. Lastly, we rank the entities by the sum of the property score from our proposed property identification method and property value scores of relevant properties.

Property Type Identification We identify property type of each relevant property $p \in P_q^*$ whether it is a qualitative or quantitative property. We use the property score of the three criteria used in Section 3.2.3.2 and also observe the majority property value type of each property. The property is considered as qualitative if the property score by the first criterion is higher than the scores by the second and third criterion. Otherwise, the property is quantitative as long as the majority property value type of the property is considered as numerical (*e.g.* date, centimeters and kilograms).

Property Value Scoring We assign the score to each property value of the relevant property for entities depending on the property type. In the case of qualitative properties, property value score can be simply assigned by the presence of the particular modifier m in the set of property values $v(e, p)$ of the property p for the entity e as defined by $s(e, m, p)$ where $s(e, m, p) = 1$ if $m \in v(e, p)$, otherwise, 0. On the other hand, we assign property value score for quantitative properties in the dissimilar way to the qualitative property. This is because the modifiers of relevant quantitative properties are associated with either ascending or descending order. We have to assign the property value based on the ranking order of the modifiers as we cannot give a high property value score to the high property value of the quantitative property if the modifier is associated with ascending order (*e.g.* the property value 2017 of the relevant quantitative property **established** of the modifier "ancient"). By this reason, we need to decide the ranking order based on the modifier by calculating means of property values of the property p from two set of entities as different set of entities will be returned by Web search engines when using different queries. Here, we use the same criteria to select two queries for the third criterion. Then, we compare the means by these two set of entities. If the mean by the original search query is larger, the modifier is associated with a descending order, otherwise an ascending order.

Afterwards, we assign the property value score to each property value v of the property p for a set of candidate entities E_q^* by the ranking order of the modifier using the equation in Table 3.1. By using these equations, the maximum property value will be assigned with the score as 1 for descending order, whereas the property value score for the maximum property value will be 0 for the ascending order.

Entity Ranking by Relevant Properties Finally, for each candidate entity $e \in E_q^*$, we rank them based on top R relevant properties for the modifier $m \in M_q$ in the query q by the sum of the property scores of the relevant properties P_q^* by the proposed property identification method in Section 3.2.3.2 and property value scores of relevant properties:

$$r(q, e) = \sum_{m \in M_q} \sum_{p \in P_q^*}^{R} z(q, m, p)\, s(e, m, p), \tag{3.13}$$

Then, we will get a list of ranked entities E_q^* based on relevant properties of modifiers indicated in the search query. Note that we can use both of the entity ranking methods described in Sections 3.2.2 and 3.2.3 by summing their ranking score, *i.e.* $c(q, e) + r(q, e)$.

3.3 Experiments

In this section, we introduce evaluation methodology for our proposed methods, and show and discuss our experimental results.

In the experiments, we used RDF data from DBpedia as a knowledge base to evaluate our methods. We obtained a set of entity types, entities, properties and property values from DBpedia, which contains structured information derived from Wikipedia. We extracted pairs of an entity and an entity type by using the predicate *is a subject of* as the property, where the subjects and objects are treated as entity types and entities, respectively. For extracting properties and property values, we retrieved all triples whose the subjects are the entities, and treated the predicates as the property names and the objects as the property values. After the extraction, we got 974,417 entity types including 4,811,226 entities and 60,252 properties from all the entities in total.

We separately evaluated resultant properties and entity ranking for measuring the performance of our proposed methods in detail. We used 50 search queries that contain modifiers and an entity type name. Within these 50 queries, we detected 62 modifiers by the method explained in Section 3.2.3.1. 34 of them correspond to qualitative properties, while the remaining correspond to quantitative properties.

3.3.1 Evaluation of Properties

This part of the evaluation was designed for evaluating the performance of our proposed property identification method. We evaluated top five properties for each modifier using: C1-C7: each of the seven criteria for identifying relevant properties, and SVM: our proposed property

	C1	C2	C3	C4	C5	C6	C7	SVM
MRR	0.447	0.225	0.171	0.172	0.038	0.066	0.041	**0.607**

Table 3.2: MRR for results of identifying relevant properties.

identification method which we combined all seven criteria using the SVM explained in Section 3.2.3.2. The first criterion alone was used as a baseline method since the existing methods (*e.g.* [17]) applied the textual similarity between a modifier and a property in order to identify the relevant property.

The relevance of each property was assessed by three assessors. They were required to judge a property as relevant only if a given modifier specifies a certain type of entity by property values of the property in a sequence. The inter-rater agreement was measured by Fleiss' Kappa [78] and was considered to be substantial at 0.735.

For the evaluation of each criterion (C1-C7), we compared the results by Mean Reciprocal Rank (MRR), which is defined as follows:

$$\frac{1}{M}\sum_{i=1}^{M}\frac{1}{r_i}, \tag{3.14}$$

where M is the number of modifiers in our experiment (*i.e.* 62), and r_i denotes the rank of the first relevant property for the i-th modifier. The MRR is often used for evaluating ranked lists where only the first relevant result matters and is suitable for this evaluation since we expect only a relevant property can be used for each modifier.

For the evaluation of the SVM, we extracted the results by the seven criteria using 62 modifiers with properties that already labeled (1 as relevance and 0 as irrelevance) as observed variables, and used the relevance of each property of each modifier as a predicted variable. In total, we obtained 877 samples. We utilized k-fold cross validation ($k = 5$ for the experiment) to evaluate our SVM model. In each fold, we computed and ranked the probabilities of the possible outcomes for predicted variables, *i.e.* the degree of relevance of each property regarding each modifier. Then, we measured the results by the MRR. We performed 10 rounds of k-fold cross validation to avoid the bias in the random sampling. As a result, we obtained the average MRR as the final measurement to determine the performance of our SVM model.

Experimental Results Table 3.2 shows the MRR achieved by seven criteria and our proposed method for identifying relevant properties. It indicates that by using our proposed method that combines all of the seven criteria using the SVM could achieve the best performance.

To drill down the results in Table 3.2, Figure 3.2 illustrates the performance in terms of the RR achieved by the first four criteria (C1-C4) and our property identification method (SVM) for each modifier, where rows represent the RR achieved by each method and columns represent the modifiers. We chose the first four criteria (C1-C4) since their MRR results suggest the highest performance among all seven in Table 3.2.

It can be seen that different criteria could identify relevant properties for different types of modifiers. For some modifiers, relevant properties

Table 3.3: Examples of properties ranked at the top by the four main criteria (C1-C4) and our property identification method (SVM). The relevant properties are the ones in red and underlined.

(a) query "ancient zen buddhist temples in kyoto"

Modifier	C1	C2	C3	C4	SVM
ancient	- - -	founded mountain -	mountain founded denomination	venerated landscape mountain	mountain country denomination
zen	denomination - -	school field works	mountain founded founder	abbot teacher landscape	denomination country lanscape
kyoto	address country -	school residence abbot	founderpriest founder address	residence country address	address country residence

(b) query "oldest architecture copenhagen"

Modifier	C1	C2	C3	C4	SVM
oldest	- - -	established length opened	client tracks architect	owned founder founded	constructionStartDate line owned

could be found at high ranks by C1, while C2, C3 and C4 could not rank them near the top (*e.g.* 21st-24th modifiers from the left). On the other hand, if relevant properties could be ranked high by C2, C3, or C4, C1 could not rank them near the top in some cases (*e.g.* 13th-15th modifiers from the left). These observations suggest that the combination of the four criteria with the other three criteria, *i.e.* all seven criteria, using the SVM could identify relevant properties most accurately by complementing each other as can be seen with the results of RR for the row representing SVM.

Table 3.3 shows examples of relevant properties. It suggests that if the modifier corresponds to qualitative properties, their properties can be successfully found by C1. If the modifier corresponds to quantitative properties, their properties can be identified by either C2, C3, or C4. These results are also consistent with our observation in Figure 3.2 that different criterion could identify relevant properties for different modifiers, and they suggest that our proposed property identification method (SVM) achieved the best performance.

3.3.2 Evaluation of Entity Ranking

To evaluate the performance of the proposed methods for returning a ranked listed of entities, we pooled top ten entities from six systems:

1. BM25: entities ranked by highest BM25 scores of their entity types.
2. WBER: entities ranked by the Web-based entity ranking.
3. PBER (1): entities ranked by the property-based entity ranking using the most relevant property ($R = 1$).
4. PBER (3): entities ranked by the property-based entity ranking using the three most relevant properties ($R = 3$).

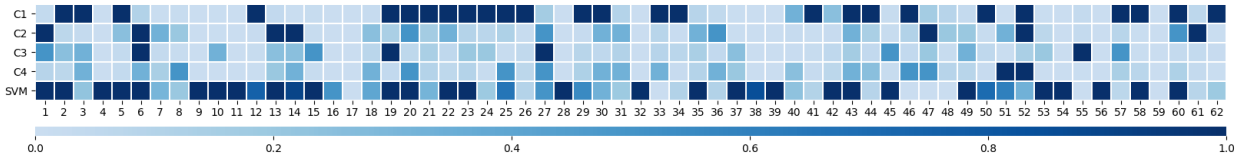


Figure 3.2: Reciprocal Rank (RR) of each of the four main criteria (C1-C4) and our property identification method which applies all seven criteria (SVM) for identifying relevant properties. Color in each cell represents the degree of RR achieved for each modifier, *i.e.* 62 modifiers in total.

	nDCG@3	nDCG@5	nDCG@10
BM25	0.459	0.487	0.529
WBER	0.600	0.628	0.692
PBER (1)	0.415	0.423	0.434
PBER (3)	0.537	0.541	0.580
WBER + PBER (1)	0.642	0.679	0.733
WBER + PBER (3)	**0.654**	**0.687**	**0.745**

Table 3.4: nDCG@3, 5, and 10 of six systems for returning a ranked list of entities using 50 queries.

5. WBER + PBER (1): entities ranked by the Web-based entity ranking and property-based entity ranking using the most relevant property ($R = 1$).
6. WBER + PBER (3): entities ranked by the Web-based entity ranking and property-based entity ranking using the three most relevant properties ($R = 3$).

The relevance of entities were assessed by the same assessors as those involved in the evaluation of properties. They were required to judge the relevance of entities for each of fifty queries based on the property value of each property with the modifiers using one of the following criteria:

- **completely relevant (4)**: property values of the entity highly correspond to all modifiers and the entity also corresponds to an entity type name in the query.
- **relevant (3)**: property values of the entity correspond to all modifiers and the entity also corresponds to an entity type name in the query.
- **somewhat relevant (2)**: property values of the entity correspond to some modifiers and the entity also corresponds to an entity type name in the query.
- **not so relevant (1)**: none of property values of the entity correspond to modifiers, but the entity corresponds to an entity type name in the query.
- **completely irrelevant (0)**: the entity does not correspond to an entity type name in the query.

Note that the assessors may find additional information if the provided property values are insufficient.

We computed the nDCG [79] as it is usually used to evaluate the effectiveness of the ranking method. $nDCG@k$ is defined as $\text{nDCG}_\text{k} = \frac{DCG_k}{idealDCG_k}$, where k indicates the number of entities, $DCG_k = \sum_{i=1}^{k} \frac{rel_i}{log_2(i+1)}$, where i is the position of the entity in the rank and rel_i denotes the relevance of the entity at position i, and $idealDCG_k$ is the ideal DCG by the relevance of k entities.

Table 3.5: Examples of top ranked entities by the six systems.

(a) query "ancient zen buddhist temples in kyoto"

BM25	**Rel**	WBER	**Rel**	PBER (1)	**Rel**
Byodo-in	1.333	Kiyomizu-dera	1.667	Kennin-ji	3.333
Historical Sites of Prince Shotoku	0	Kinkaku-ji	3.0	Tenryu-ji	3.333
Sanzen-in	1.333	Ninna-ji	2.0	Kinkaku-ji	3.0
PBER (3)	**Rel**	WBER + PBER (1)	**Rel**	WBER + PBER (3)	**Rel**
Tenryu-ji	3.333	Kinkaku-ji	3.0	Kinkaku-ji	3.0
Kennin-ji	3.333	Kiyomizu-dera	1.667	Kennin-ji	3.333
Ginkaku-ji	3.0	Kennin-ji	3.333	Kiyomizu-dera	1.667

(b) query "oldest architecture copenhagen"

BM25	**Rel**	WBER	**Rel**	PBER (1)	**Rel**
Architecture in Copenhagen	0.0	Amalienborg	3.0	Tøjhus Museum	2.667
COBE Architects	0.333	Rosenborg Castle	4.0	Ny Carlsberg Glyptotek	2.333
Rørbæk & Møller Arkitekter	0.333	Christiansborg Palace	2.333	National Gallery of Denmark	2.0
PBER (3)	**Rel**	WBER + PBER (1)	**Rel**	WBER + PBER (3)	**Rel**
Henning Larsen Architects	0	Amalienborg	3.0	Amalienborg	3.0
Tegnestuen Vandkunsten	0	Rosenborg Castle	4.0	Rosenborg Castle	4.0
Tøjhus Museum	2.667	Christiansborg Palace	2.333	Christiansborg Palace	2.333

Experimental Results Table 3.4 shows the performance of entity ranking in terms of the nDCG@3, 5, and 10 using 50 queries from six systems. The results suggest that entities ranked by the combination of the Web-based entity ranking and property-based entity ranking using top three relevant properties (WBER + PBER (3)) achieved the highest nDCG@3, 5 and 10, while PBER (1) achieved the lowest. For WBER, its ranking performance was good, but it is still not better than the performance of (WBER + PBER (1)) and (WBER + PBER (3)). However, WBER beats the performance of the system BM25 that simply used the similarity score between the query with the entity type names from a knowledge base since we also applied the document frequency of the entities from Web search results using the search query. Moreover, by using the property-based entity ranking, that involves the ranked list of entities by Web-based entity ranking with the relevant properties for the modifiers in the query (WBER + PBER (1) and WBER + PBER (3)), improves the ranking performance as this system explicitly concerns about the modifiers. Ranking by the relevant properties of the modifiers affects the better entity ranking results because it directly distinguishes the entities by their property values of the relevant properties (*i.e.* entity-related information), which the functionality of Web search engine cannot handle and process the modifiers in a search query effectively. However, we cannot use the property-based entity ranking without the Web-based entity ranking method as the performance results of PBER (1) and PBER (3) suggested. This is because the Web-based entity ranking helps finding the entities that would correspond to the entity type name in the query. Without this method, we could get the entities that relevant to the modifiers, but not relevant to the entity type name in the query.

Table 3.5 shows examples of entities ranked by six systems for two queries. The results suggest that WBER + PBER (3) could achieve and get the relevant entities ranked near the top of the list for both queries.

3.4 Summary

In this study, we proposed methods of returning a ranked list of entities for a given query by leveraging different types of modifiers. Our first proposed method, *i.e.* the Web-based entity ranking, involves the similarity between a query and entity type names together with the frequency of entities in search results returned in response to the query. Moreover, we proposed the property-based entity ranking which includes the part of identifying a property corresponding to each modifier in a query in which we proposed the property identification method based on the combination of seven different criteria using SVM. Experimental results showed that our property identification method could identify more relevant properties, and the combination of our Web-based and property-based entity ranking methods could return more relevant entities at the top rank.

Identifying Entity Properties from Text with Zero-Shot Learning

4

4.1 Introduction

Entities play an important role in modern search engines due to the significant number of entity-related queries [1–3] 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 [80].

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 [10–13]. 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 are 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 [52]). 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.

4.2 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.

4.2.1 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.

4.2.2 Model

Our proposed model comprises five main components (Figure 4.1): (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.

4.2.2.1 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

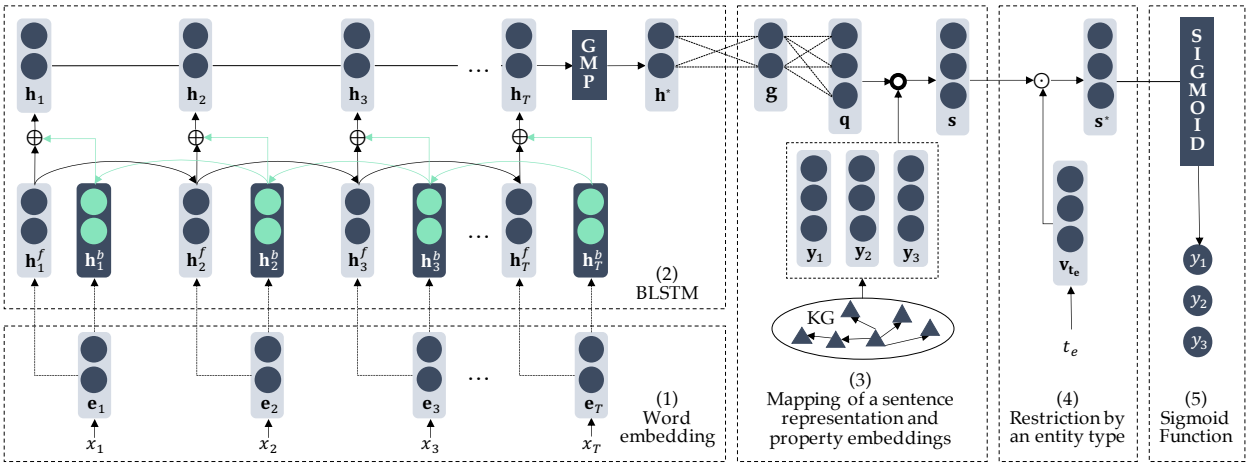


Figure 4.1: Model architecture: (1) a word embedding that maps words with low-dimensional vectors, (2) a bidirectional long short-term memory that generates a sentence-level representation, (3) a mapping that learns the transformation between representations of sentences and properties, (4) an entity type that gives the emphasis on the properties that are likely to be represented by the sentence, and (5) a sigmoid function that produces probabilities over all properties.

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 $\mathbf{e}_t = \mathbf{E_w}\mathbf{v}_t$.

4.2.2.2 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 [81]. 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:

$$\mathbf{i}_t = \sigma\left(\mathbf{W_i}\left[\mathbf{h}_{t-1}, \mathbf{e}_t\right] + \mathbf{b_i}\right) \tag{4.1}$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W_f}\left[\mathbf{h}_{t-1}, \mathbf{e}_t\right] + \mathbf{b_f}\right) \tag{4.2}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W_o}\left[\mathbf{h}_{t-1}, \mathbf{e}_t\right] + \mathbf{b_o}\right) \tag{4.3}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{W_c}\left[\mathbf{h}_{t-1}, \mathbf{e}_t\right] + \mathbf{b_c}\right) \tag{4.4}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right) \tag{4.5}$$

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 [82], 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.

4.2.2.3 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 4.2.4. 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 including non-linear activation (*i.e.* a rectified linear unit (ReLU)):

$$\mathbf{g} = \text{ReLU}(\mathbf{W_g}\mathbf{h}^* + \mathbf{b_g}) \tag{4.6}$$

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 the mapping between the sentence representation with d_h dimensions into the property embeddings with d_y dimensions as follows:

$$\mathbf{q} = \mathbf{W_q}\mathbf{g} + \mathbf{b_q} \tag{4.7}$$

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 $\mathbf{s}$ as follows:

$$\mathbf{s} = \mathbf{E_y}\mathbf{q} \tag{4.8}$$

As illustrated in Figure 4.2, 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.

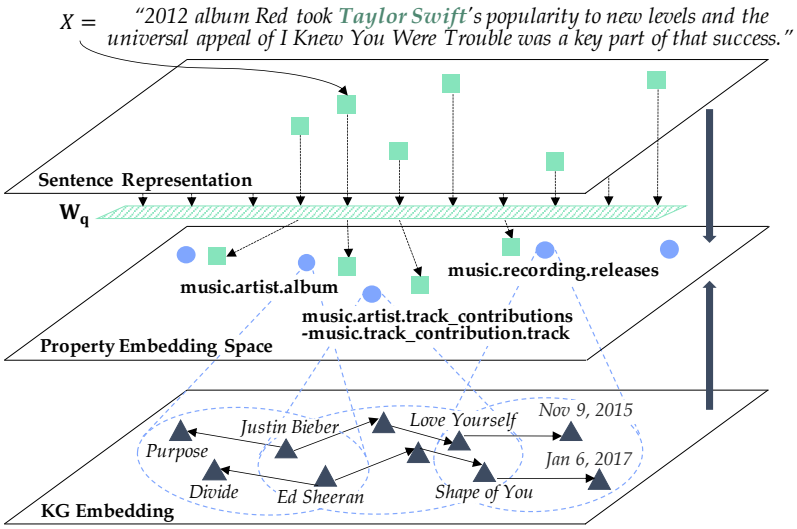


Figure 4.2: Mapping between sentence representation and property embeddings.

4.2.2.4 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 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 $\mathbf{v_{t_e}} \in \{0,1\}^{|Y'|}$ 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 $\mathbf{v_{t_e}}$ is used to perform element-wise multiplication using the output from the previous component as follows:

$$\mathbf{s}^* = \mathbf{s} \odot \mathbf{v_{t_e}} \tag{4.9}$$

In this way, the entity type vector suppresses prediction of irrelevant properties.

4.2.2.5 Sigmoid function

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

4.2.3 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:

$$L_{\text{CLF}} = -\sum_{i}\sum_{j} y_{i,j} \log \hat{y}_{i,j} \tag{4.10}$$

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 [83] as our optimization algorithm.

4.2.4 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 [52].

4.2.4.1 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.

4.2.4.2 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,

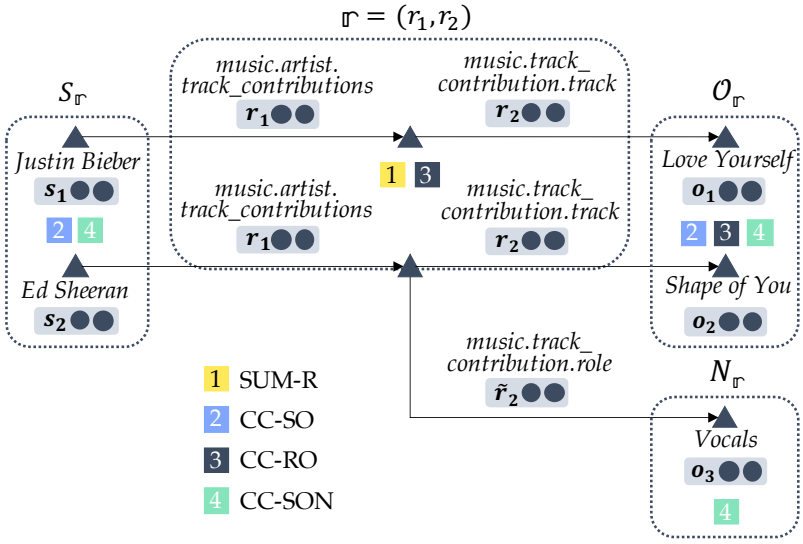


Figure 4.3: Example components used in each of the four techniques to represent property embeddings using KG embedding. This assumes that the property is derived from a predicate path, $\mathbb{r}$, which comprising $n(=2)$ predicates.

$\mathbf{o}$, such that $\mathbf{s} + \mathbf{r} \approx \mathbf{o}$. Such embeddings are learned by minimizing the margin-based ranking loss:

$$L_{\text{KG}} = \sum_{(s,r,o)\in\mathcal{K}} \sum_{(s',r,o')\in\mathcal{K}'} [\gamma + d(\mathbf{s}+\mathbf{r},\mathbf{o}) - d(\mathbf{s}'+\mathbf{r},\mathbf{o}')]_+ \tag{4.11}$$

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.

4.2.4.3 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 4.3). 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, $[\mathbf{s}, \mathbf{o}]$. 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, $[\mathbf{r_1}, \mathbf{o}]$. 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, $[\mathbf{s}, \mathbf{o}, \mathbf{n}]$. 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.

The components for these techniques are computed as follows:

$$\mathbf{s} = \frac{\sum_{s \in S_{\mathbb{r}}} \mathbf{s}}{|S_{\mathbb{r}}|} \tag{4.12}$$

$$\mathbf{o} = \frac{\sum_{o \in \mathcal{O}_{\mathbb{r}}} \mathbf{o}}{|\mathcal{O}_{\mathbb{r}}|} \tag{4.13}$$

$$\mathbf{n} = \begin{cases} \frac{\sum_{o \in N_{\mathbb{r}}} \mathbf{o}}{|N_{\mathbb{r}}|} & (n > 1) \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{4.14}$$

where $S_{\mathbb{r}} = \{s \,|\, (s, \mathbb{r}, o) \in \mathcal{P}\}, \mathcal{O}_{\mathbb{r}} = \{o \,|\, (s, \mathbb{r}, o) \in \mathcal{P}\}, N_{\mathbb{r}} = \{o \,|\, (s, (r_1, r_2, \cdots, r_{n-1}, \tilde{r}_n), o) \in \mathcal{P}\}$, $\tilde{r}_n$ represents a neighbor predicate of the predicate r_n, and n denotes the predicate path length.

4.3 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.

4.3.1 Datasets

We used two datasets to evaluate our model. The first is a widely used dataset (NYT10) originally released by Riedel *et al.* [11] 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.

4.3.1.1 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.* [52] and was used to generate the KG embedding for our classification model. Details of the FB15k dataset are reported in Section 4.3.3.2. 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.

1: https://developers.google.com/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.

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

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 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.

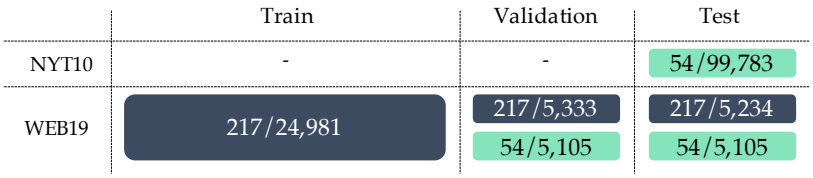


Figure 4.4: Number of properties and sentences in the train, validation, and test sets for the NYT10 and WEB19 datasets. The dark blue and green boxes indicate sets of sentences for seen and unseen properties, respectively.

We computed a Free-Marginal Multirater Kappa [84] 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 4.4 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.

4.3.2 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).

4.3.2.1 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:

$$\mathrm{A}_{\text{sample}} = \frac{|Y_s \cap \hat{Y}_s|}{|Y_s \cup \hat{Y}_s|} \quad (4.15) \qquad \mathrm{P}_{\text{sample}} = \frac{|Y_s \cap \hat{Y}_s|}{|\hat{Y}_s|} \quad (4.16)$$

$$\mathrm{R}_{\text{sample}} = \frac{|Y_s \cap \hat{Y}_s|}{|Y_s|} \quad (4.17) \qquad \mathrm{F}_{\text{sample}} = \frac{2|Y_s \cap \hat{Y}_s|}{|Y_s| + |\hat{Y}_s|} \quad (4.18)$$

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.

Table 4.1: 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

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.

4.3.2.2 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: $F_{label} = \frac{2|S_l \cap \hat{S}_l|}{|S_l|+|\hat{S}_l|}$, 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.

4.3.3 Experimental Settings

4.3.3.1 Word embedding

We trained a Skip-gram model [77] on the English Wikipedia articles[3] . In the experiments, we used word embeddings with 100, 200, and 300 dimensions.

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

4.3.3.2 KG embedding

To obtain the KG embedding, we relied on the original dataset (FB15k) published by Bordes *et al.* [52] 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, $\curvearrowright$, 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 [52] by setting all parameters for training the KG embedding to be the same as the optimal settings for the FB15k dataset.

4.3.3.3 Comparison of the proposed models and baseline methods

We compared our proposed neural network-based models with baseline methods while considering variations of our proposed model. 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 [40].
- **CNN**: CNN model [41].
- **BRNN**: Bidirectional RNN model [42].
- **LSTM**: LSTM model.
- **BLSTM**: Bidirectional LSTM model.

Table 4.2: 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 ($\mathbf{F_{label}}$).

		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

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.

4.3.4 Results

4.3.4.1 RQ1 - *How effectively can our proposed model handle the ZSL problem relative to baseline methods?*

Table 4.1 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

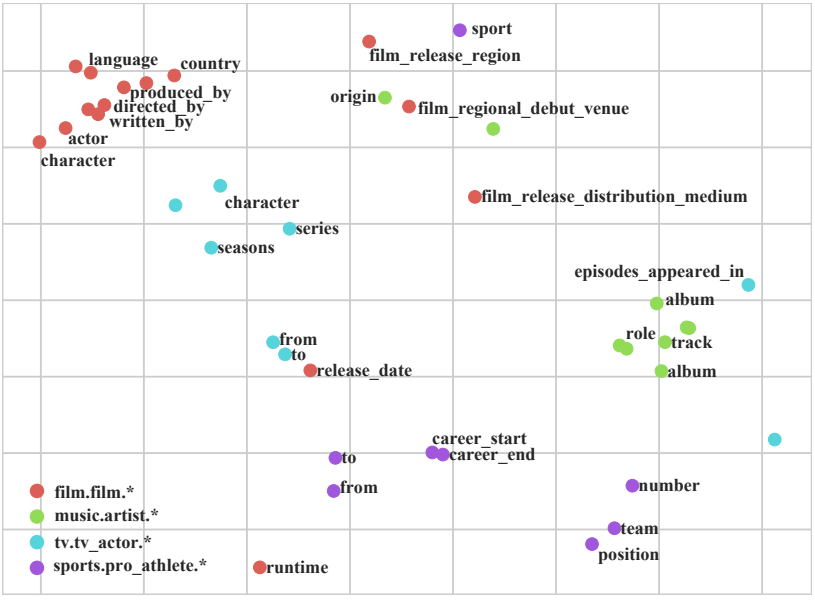


Figure 4.5: The embeddings of properties constructed by SUM-R technique of the four entity types on the WEB19 dataset comprising a large number of properties on the embedding space.

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 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.

4.3.4.2 RQ2 - *Which technique(s) is the best for representing properties with embeddings in our case?*

The experimental results on Table 4.1 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 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 [85], Figure 4.5 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.

4.3.4.3 RQ3 - *Under what situation(s) can the model work effectively for handling the ZSL problem?*

Figure 4.6 illustrates the relative performance in identifying unseen properties on the WEB19 dataset using the best proposed model #6, as described in Table 4.1, 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

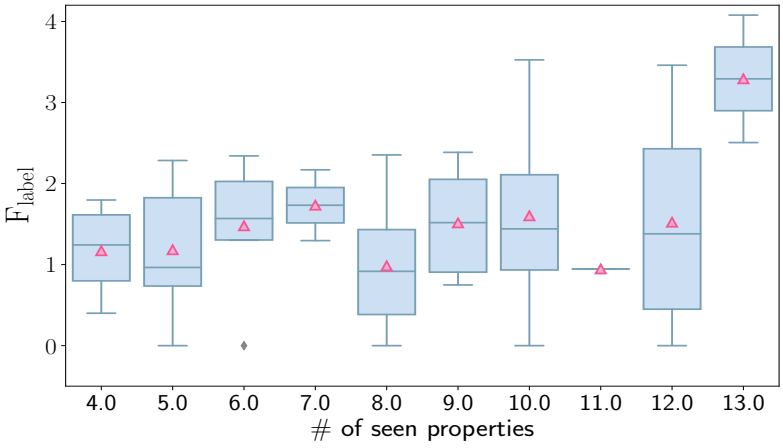


Figure 4.6: Relative performance in identifying unseen properties on the WEB19 dataset of the best proposed model #6 that utilized the entity type and the property embeddings constructed by SUM-R technique as compared with the best baseline method (Rand-T) in terms of label-based F-measure ($\mathbf{F_{label}}$) relative to the number of seen properties of the same entity type.

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 4.2. 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.

4.4 Summary

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.

Explaining Relationship of Entity Sets

5

5.1 Introduction

With a vast amount of entities over the Web and news articles, user's knowledge for interpreting a relationship between entities is undeniably important to understand the connections and circumstances of entities behind each article since such article mostly mentions multiple entities, but does not always explain the background knowledge of those entities in general. For example, Figure 5.1(a) illustrates a short section of an article involving four entities including "The Lion King", "Tim Rice", "Elton John" and "Disney". With zero or limited knowledge regarding those entities, the users might find it difficult to understand the whole context of this article. Thus, incorporating additional information such as a textual explanation that describes how all entities are related, as shown in Figure 5.1(b), would enable the users to gain an insight about the relationship between entities and, as a result, be able to infer the meaning or intention behind each article.

The problem of explaining a relationship between entities using text has previously been addressed in some existing works [14–16]. However, their approaches were specifically designed for explaining a pair of entities based on a particular relationship (*i.e.* a predicate) in a Knowledge Graph (KG). When considering more than two entities, the existing approaches might not effectively be applied as the relationship of a set of entities is represented by multi-hop paths where each path consists of a different sequence of entities and predicates such as (Tim Rice, *lyricist.lyrics_written*, Can You Feel the Love Tonight, *recording.artist*, Elton John, *recording.song*, Can You Feel the Love Tonight, *film.music*, The Lion King, *film.production_companies*, Disney). Thus, the reasoning over the KG is necessary to determine the informative path that could lead to a more descriptive relationship explanation.

To explain a relationship of a set of entities, we propose a learning framework that fully utilizes a structure and components of the KG for generating a textual relationship explanation for a set of entities, namely "XERG" which stands for "eXplaining Entity Relationship with knowledge Graph". Our XERG consists of two main components including: 1) the Entity Relationship Reasoning (ERR) network that learns to explore an informative path between given entities through a reinforcement learning (RL) and 2) the encoder-decoder network that takes as an input the reasoning path and generates a textual relationship explanation as an output along with feedback that informs the ERR network about the informativeness of the path according to the generated explanation. These two components are jointly learned and dependent on each other as the ERR network learns to adjust the reasoning based on the signals from the encoder-decoder network, while the encoder-decoder network learns to generate the explanation based on the path received from the ERR network.

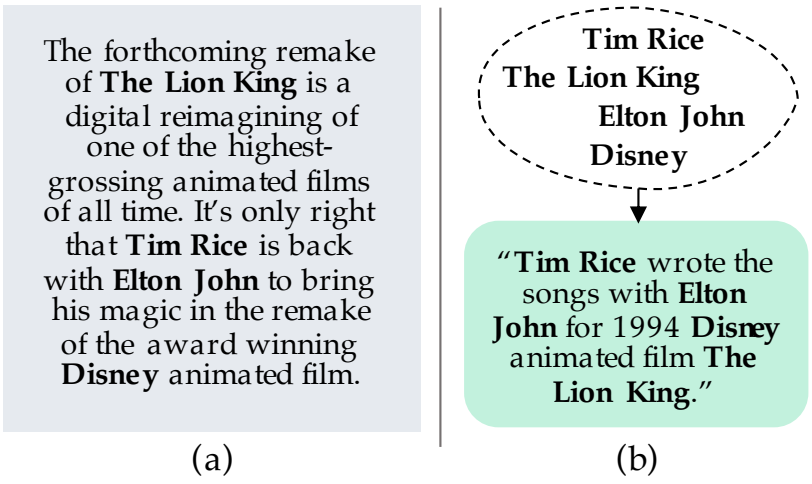


Figure 5.1: (a) an example article that mentions multiple entities and (b) a textual relationship explanation for the entities appearing in the article.

In the experiments, we utilized our newly constructed dataset to evaluate our proposed learning framework and baselines. We found that our learning framework outperformed every baseline for most evaluation metrics in terms of providing the relationship explanations for the entity sets.

Our contributions are summarized as follows:

- We aimed at a generation of a textual relationship explanation for an entity set by fully utilizing the KG structure and its information.
- We introduced a learning framework (XERG) consisting of the ERR and encoder-decoder networks which are trained together in an end-to-end fashion.
- We built a new dataset for comparing the performance of our proposed framework with baselines.

5.2 Methodology

We introduce our learning framework for generating a textual explanation for a relationship of a set of entities in this section.

5.2.1 Problem Formulation

Given a set of entities E, our problem is to predict a sequence of tokens where each token can be represented by either a vocabulary $v \in V$ or an entity name $n(e)$ where $e \in E$.

In the following subsections, we describe the details of the two components in our learning framework including: 1) the Entity Relationship Reasoning (ERR) network that explores an informative path p of a given E within the KG and 2) the encoder-decoder network that generates a relationship explanation based on p as an output along with feedback to the ERR network.

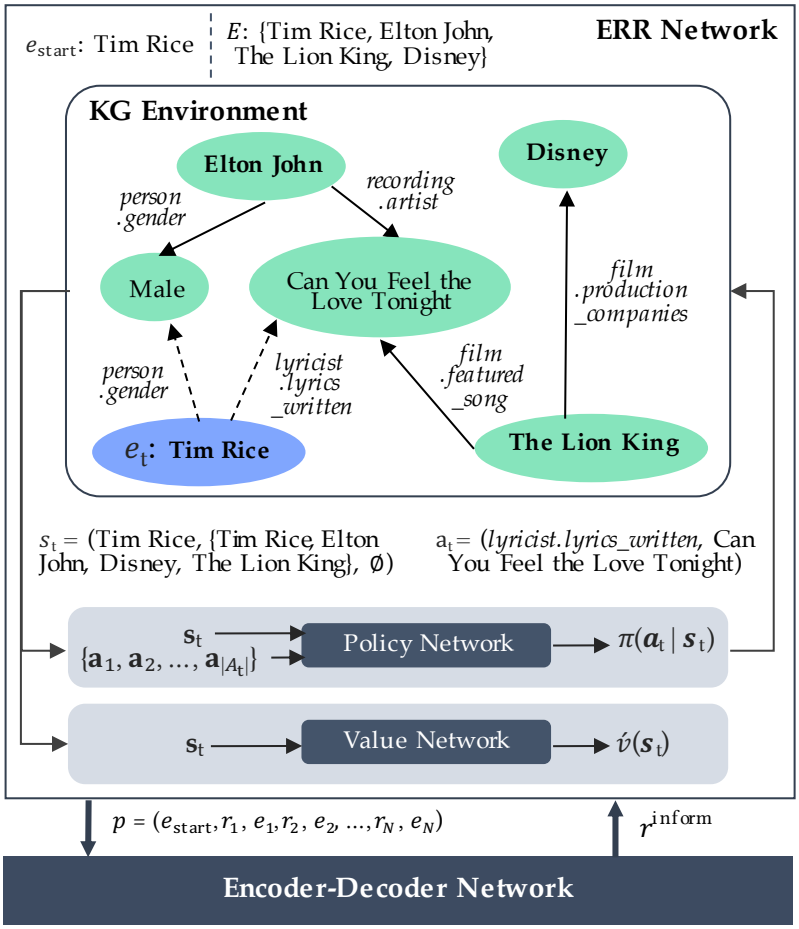


Figure 5.2: Entity Relationship Reasoning Network.

5.2.2 Entity Relationship Reasoning Network

Taken as the input a set of entities E, the main objective of this component is to find a path p consisting of E using a subgraph of the KG including E, $\mathcal{G}_E$, through the RL. The path p is represented by a sequence of entities and predicates, $(e_{\text{start}}, r_1, e_1, r_2, e_2, \ldots, r_N, e_N)$, where e_{start} denotes a start entity and N denotes a number of predicates connected to entities in p.

To achieve this goal, we rely on a policy gradient method, REINFORCE with baseline [86]. The fundamental idea of this method is that a policy agent learns to traverse along $\mathcal{G}_E$ by selecting a predicate via a *policy network*, and at the same time, the policy agent learns to improve its selection based on the learned value from a *value network* as a baseline that evaluates the selected predicate at each time step to maximize rewards. Figure 5.2 illustrates the overall mechanism of the ERR network.

To encourage the policy agent to find paths that include every entity in E and, more importantly, are useful for generating descriptive explanation, we consider two criteria as rewards based on the completeness of the path p whether it contains every entity in E and the informativeness of p according to the generated explanation given p.

5.2.2.1 Reinforcement Learning Formulation

We define $\mathcal{G}_E = \{(e_s, r, e_o) \mid e_s, e_o \in \mathcal{E}, r \in \mathcal{R}, E \subset \mathcal{E}\}$, where $\mathcal{E}$ and $\mathcal{R}$ represent a set of entities and predicates respectively. Each predicate r of each triplet, (e_s, r, e_o), is a directed edge between an entity subject e_s and an entity object e_o.

We define the reasoning of paths as Markov Decision Process (MDP) [86] which consists of the following components:

States Each state s_t is defined as (e_t, E, E_v), where e_t denotes an entity e_t at step t and E_v denotes the entities in E that were already visited by the agent. We incorporate E_v to represent the state because it could inform the agent of which entities that it already visited such that the agent is encouraged to discover other entities in E.

Actions A set of possible actions A_t at time t consists of every pair of an outgoing edge r from the current entity e_t and an entity $e_{t'}$ linked by r, *i.e.* $A_t = \{(r, e'_t) \mid (e_t, r, e'_t) \in \mathcal{G}_E\}$.

Transition Given a state $s_t = (e_t, E, E_v)$ and an action $a_t = (r, e_{t'})$, the transition function is defined as $s_{t+1} = T(s_t, a_t) = (e_{t'}, E, E_v^{e_{t'}})$.

Rewards Two criteria to determine rewards (*i.e.* r^{comp} that measures path completeness and r^{inform} that measures path informativeness) are defined as follows:

$$r^{\text{comp}} = \begin{cases} 1, & \text{if } e_t \in E \text{ and } e_t \notin E_v \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

$$r^{\text{inform}} = f(S_E, \hat{S}_E) \tag{5.2}$$

where $f(\circ)$ denotes an evaluation metric measuring the quality of a generated explanation S_E with a ground truth relationship explanation, $\hat{S}_E$ of E. Note that r^{inform} is only given at a terminal state where every entity in E was reached by the agent.

Then, we combine the two criteria as a reward $r_t = \alpha_r(r^{\text{comp}}) + \beta_r(r^{\text{inform}})$, where α_r and β_r are hyperparameters controlling the degrees of the two criteria ranging from 0 to 1.

5.2.2.2 Policy Network

The policy network $\pi(\mathbf{a}_t|\mathbf{s}_t)$ takes as the input a state vector $\mathbf{s}_t$ and a vector of each possible action $\mathbf{a}_i$ where $a_i \in A_t$. Here, we represent $\mathbf{s}_t$ by $[\mathbf{e}_t, \mathbf{E}, \mathbf{E_v}]$ which indicates a concatenation of an embedding of an entity e_t ($\mathbf{e}_t$), an average of embeddings of entities in E ($\mathbf{E}$), and an average of embeddings of E_v ($\mathbf{E_v}$), while each action vector $\mathbf{a}_i$ is represented by $[\mathbf{r}_i^{e_t}, \mathbf{e}_i^{e_t}]$ where $\mathbf{r}_i^{e_t}$ denotes an embedding of each outgoing edge (*i.e.* a

predicate) from e_t, and $\mathbf{e}_i^{e_t}$ denotes an embedding of each entity linked by $r_i^{e_t}$ from e_t. The policy network is defined as follows:

$$\pi(\mathbf{a}_t|\mathbf{s}_t) = \frac{\exp(b_{t,i})}{\sum_i^{|A_t|}\exp(b_{t,i})} \tag{5.3}$$

$$b_{t,i} = \mathbf{W}_2^{\pi}(\text{ReLU}(\mathbf{W}_1^{\pi}[\mathbf{s}_t;\mathbf{a}_i])) \tag{5.4}$$

where $\mathbf{W}_1^{\pi} \in \mathbb{R}^{(d_s+d_a)\times d_h}$ and $\mathbf{W}_2^{\pi} \in \mathbb{R}^{d_h\times 1}$ are weight matrices, where d_s, d_a, d_h are dimensions of a state, action and hidden vectors, and ReLU is a non-linear function, *i.e.* a Rectified Linear Unit.

5.2.2.3 Value Network

The value network $\acute{v}(\mathbf{s}_t)$ takes a state vector $\mathbf{s}_t$ as an input to produce a real value that instructs the agent of how good it is to be in s_t according to the previously selected action. The value network is defined as follows:

$$\acute{v}(\mathbf{s}_t) = \mathbf{W}_2^{\acute{v}}(\text{ReLU}(\mathbf{W}_1^{\acute{v}}\mathbf{s}_t)) \tag{5.5}$$

where $\mathbf{W}_1^{\acute{v}} \in \mathbb{R}^{d_s\times d_h}$ and $\mathbf{W}_2^{\acute{v}} \in \mathbb{R}^{d_h\times 1}$ are weight matrices.

5.2.2.4 Learning

Letting $\theta = \{\mathbf{W}_1^{\pi}, \mathbf{W}_2^{\pi}, \mathbf{W}_1^{\acute{v}}, \mathbf{W}_2^{\acute{v}}\}$ represents parameters of the policy and value networks, we optimize the policy gradient as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log\pi(\mathbf{a}_t|\mathbf{s}_t)(R - \acute{v}(\mathbf{s}_t))] \tag{5.6}$$

where R denotes the discounted cumulative reward from state s_t to the terminal state.

For training the ERR network, we adopted Adam [83] as our optimizer.

5.2.3 Encoder-Decoder Network

Figure 5.3 illustrates the overall mechanism of the encoder-decoder network in which it receives a reasoning path p from the ERR network to generate a sequence of tokens Y where each token y_t can either be a vocabulary $v \in V$ or an entity names $n(e)$ where $e \in E$. Then, a reward, r^{inform}, is computed between the generated and ground truth explanations and will be passed as feedback to the ERR network.

For the decoder, we incorporate the attention mechanism (ATTN) that learns to highlight the relevant part of the input path which, in our case, is either the entity or the predicate. Additionally, we introduce a Word-Entity Generator (WEG) that alternately outputs a vocabulary or an entity name.

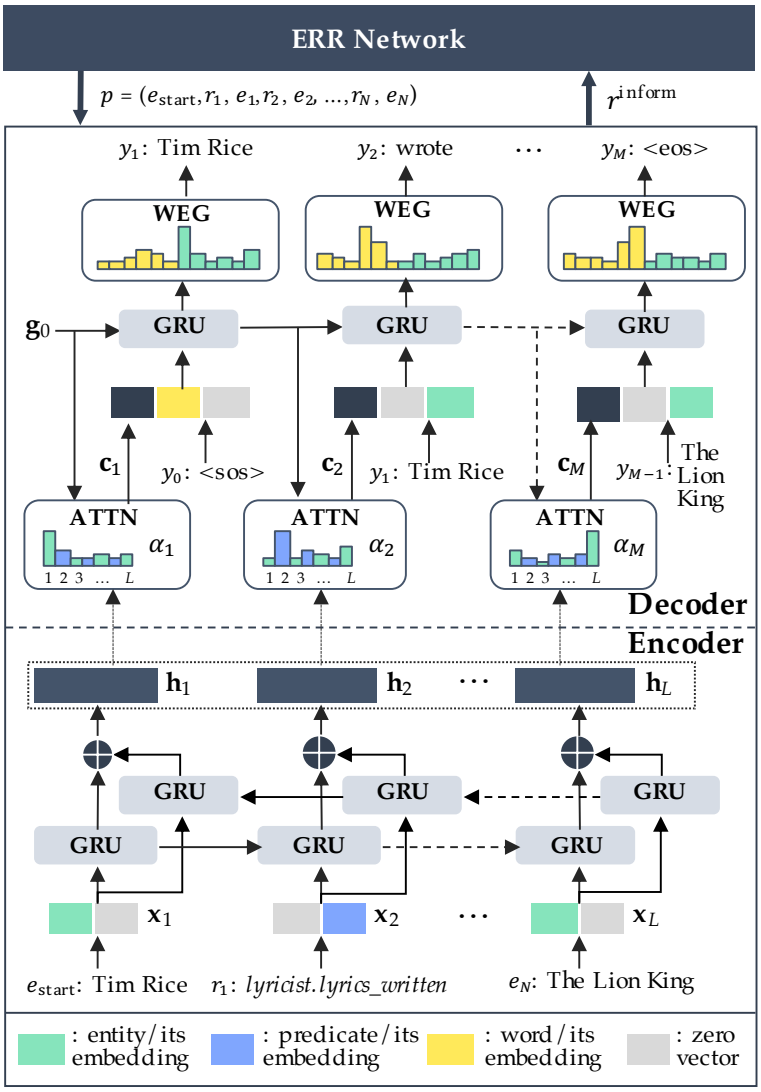


Figure 5.3: Encoder-Decoder Network.

5.2.3.1 Encoder

The encoder receives a path $p = (e_{start}, r_1, e_1, r_2, e_2, \ldots, r_N, e_N)$ as an input with length $L = 2N + 1$ and generates a path representation via a bi-directional Gated Recurrent Unit (GRU) [87]. Note that, for p that does not include every $e \in E$, we fill the path with pairs of < pad > and each remaining entity.

Since the path p contains a sequence of entities and predicates, we represent each element $x_t \in p$ by an embedding $\mathbf{x}_t$ as $[\mathbf{e}_t; \mathbf{0}_r]$ for an entity e_t and $[\mathbf{0}_e; \mathbf{r}_t]$ for a predicate r_t, where $\mathbf{e}_t$ is an embedding of e_t, $\mathbf{r}_t$ is an embedding of r_t, $\mathbf{0}_e$ is a zero vector with a dimension of $\mathbf{e}_t$, and $\mathbf{0}_r$ is a zero vector with the dimension of $\mathbf{r}_t$.

Each hidden vector $\mathbf{h}_t$ is obtained by an element-wise addition between hidden vectors from forward ($\overrightarrow{\mathbf{h}_t}$) and backward ($\overleftarrow{\mathbf{h}_t}$) layers of the GRU where the hidden vector from each direction is derived by $\text{GRU}(\mathbf{h}_{t-1}, \mathbf{x}_t)$.

Thus, the last hidden vector $\mathbf{h}_L$ is regarded as the representation of p and used as the first hidden vector $\mathbf{g}_0$ for the decoder.

5.2.3.2 Decoder

Upon receiving a path representation from the encoder network as the first hidden vector for the decoder, the decoder generates a sequence of tokens $Y = (y_1, y_2, \ldots, y_M)$ where each y_t can be either vocabularies or entity names. To represent each token y_t by an embedding $\mathbf{y}_t$, we utilize $[\mathbf{v}_t; \mathbf{0}_e]$ when y_t indicates the vocabulary v_t, while utilize $[\mathbf{0}_v; \mathbf{e}_t]$ when y_t indicates the entity name of the entity e_t, where $\mathbf{v}_t$ is an embedding of v_t, $\mathbf{e}_t$ is an embedding of e_t, $\mathbf{0}_v$ is a zero vector with a dimension of $\mathbf{v}_t$, and $\mathbf{0}_e$ is a zero vector with the dimension of $\mathbf{e}_t$.

At each decoding step t, we first compute a context vector $\mathbf{c}_t$ via the ATTN as follows:

$$\mathbf{c}_t = \sum_i^L \alpha_{t,i} \mathbf{h}_i \tag{5.7}$$

$$\alpha_{t,i} = \frac{\exp(\beta_{t,i})}{\sum_i^L \exp(\beta_{t,i})} \tag{5.8}$$

$$\beta_{t,i} = \mathbf{W}_3^a(\text{ReLU}(\mathbf{W}_2^a \mathbf{g}_{t-1} + \mathbf{W}_1^a \mathbf{h}_i])) \tag{5.9}$$

where $\mathbf{W}_1^a, \mathbf{W}_2^a \in \mathbb{R}^{d_h \times d_h}$ and $\mathbf{W}_3^a \in \mathbb{R}^{d_h \times 1}$ are weight matrices.

After that, a hidden vector $\mathbf{g}_t$ at each step t is computed using the GRU as $\text{GRU}(\mathbf{g}_{t-1}, [\mathbf{c}_t; \mathbf{y}_{t-1}])$.

According to each $\mathbf{g}_t$, we utilize the WEG to produce distributions of the vocabularies $P_v(y_t = v)$ and those of the entity names $P_e(y_t = n(e))$ as follows:

$$P_v(y_t = v) = \text{softmax}(\mathbf{W}^v \mathbf{g}_t) \tag{5.10}$$

$$P_e(y_t = n(e)) = \frac{\exp(\beta_{t,i})}{\sum_i^{|E|} \exp(\beta_{t,i})} \tag{5.11}$$

where $\mathbf{W}^v \in \mathbb{R}^{d_h \times |V|}$ denotes a weight matrix, softmax represents a softmax function, and $\beta_{t,i}$ is computed by $\mathbf{g}_t^\intercal \mathbf{W}^e \mathbf{e}_i$, where $\mathbf{W}^e \in \mathbb{R}^{d_e \times d_h}$ is a weight matrix.

Additionally, the WEG measures a score $\gamma_t \in [0, 1]$ that determines how likely each output token y_t should be a vocabulary or an entity name through a fully connected layer with a sigmoid function as $\gamma_t = \text{sigmoid}(\mathbf{W}^\gamma \mathbf{g}_t)$, where $\mathbf{W}^\gamma \in \mathbb{R}^{d_h \times 1}$ is a weight matrix.

Each γ_t score is then used to penalize the distribution of vocabularies and the distribution of the entity names as follows:

$$\mathbf{z}_t \sim P(y_t) = \begin{bmatrix} (1 - \gamma_t) P_v(y_t = v) \\ \gamma_t P_e(y_t = n(e)) \end{bmatrix} \tag{5.12}$$

In this way, if γ_t score is high, it is more likely that the output token should be an entity name, otherwise, the output token should be a vocabulary.

5.2.3.3 Learning

For training the encoder-decoder network, we combine two losses, $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$, based on cross-entropy as our objective function. $\mathcal{L}_1$

is computed using the target $\mathbf{z}$ and predicted $\mathbf{z}$ probability distribution as $-\sum_t^M \mathbf{z}_t \log(\mathbf{z}_t)$. On the other hand, $\mathcal{L}_2$ is computed between the target $\hat{\gamma} \in \{0, 1\}$ and predicted γ scores which determine the positions of the vocabularies and entity names in the sequence of tokens as $-\sum_t^M (\hat{\gamma}_t \log(\gamma_t) + (1 - \hat{\gamma}_t) \log(1 - \gamma_t))$, where M denotes the length of a generated sequence of tokens.

We trained the encoder-decoder network using Adam [83] as the optimizer.

5.3 Experiments

We describe experiments on our dataset which was constructed for our research problem to evaluate our proposed learning framework, XERG. We aim to address the following research questions:

- **RQ1**: Do the information within the KG and its structure help generating better relationship explanations of entities?
- **RQ2**: Which criterion for determining the rewards (r^{comp} and r^{inform}) for the entity relationship reasoning is more effective for generating the explanations?

5.3.1 Dataset

Since there is no publicly available dataset suitable for our research problem, we constructed our own dataset consisting of sets of entities where each set E involves a ground truth explanation of a relationship of entities as well as a subgraph of the KG including E, $\mathcal{G}_E$.

To obtain the dataset, we first selected top 10% of popular Wikipedia articles using their total number of pageviews from October 1, 2019, to December 31, 2019, of each of the six categories including: 1) person, 2) organization, 3) film, 4) location, 5) event, and 6) invention. Next, for each article, we extracted a set of entities from a sentence that its length is less than 50 and kept only the entity set with the number of entities between two and five. Additionally, we applied two types of enrichments on the sentences. First, by following [88], we replaced pronouns in each sentence of each article with the title of the article's entity based on the number of occurrences between the pronouns "he" and "she", if the article's entity does not already appear in the sentence. Second, we removed the time occurrences such as "in 1992" and "on 26 June 2003" from the sentences since we do not focus on the numerical information around entities in this work.

After that, by utilizing Freebase[1] as our KG, we obtained $\mathcal{G}_E$ by collecting paths connecting each pair of entities in the set with a limited number of predicates equals to three and combining a set of triplets from every path as a subgraph. However, to avoid ending up with too massive graphs, we ignored predicates that indicate the types of entities since they could lead us to have too many paths for each entity pair. Then, we omitted the entity sets that we could not find any path connecting every pair of entities to ensure the connectivity of all entities. Moreover, two additional types of edges between the entities are added to allow better graph exploration

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

as follows: 1) an inverse directed edge for each predicate that links the entity object to the entity subject of any triplet in $\mathcal{G}_E$ and 2) a self-loop edge that links any entity in $\mathcal{E}$ to itself. Later, for every remaining entity set, we set each article's entity as e_{start} for finding the path p.

We ended up having 168,273 entity sets with 138,666 unique entities in total. We separated the entity sets into 134,618, 16,827, and 16,828 for training, validation, and testing, respectively.

5.3.2 Evaluation Metrics

We considered multiple evaluation metrics including BLEU [89], ROUGE-L [90] and METEOR [91] to measure the quality of the generated relationship explanation.

5.3.3 Experimental Settings

5.3.3.1 ERR Network

We relied on TransE [52], which is currently among the popular KG embedding techniques, to represent entities and predicates by the embeddings. We trained the embeddings using a fast implementation of TransE by lin2015learning[2] on the triplets from all the subgraphs of the entity sets in our newly constructed dataset. We opted to use the entity and predicate embeddings with 100 dimensions in the experiments.

2: https://github.com/thunlp/Fast-TransX

We set all hidden vector dimensions used in this network to 300 and set a maximum number of predicates for each reasoning over the KG to 20.

5.3.3.2 Encoder-Decoder Network

We prepared a set of vocabularies based on the ground truth explanations in our dataset by converting every token into a lower case and accumulating only a vocabulary that appears at least five times across every explanation. For those vocabularies appearing less than five times, we replaced them with < unk > tokens. As a result, we get a set of vocabularies of size 19,296.

We utilized pre-trained word embeddings[3] which were trained by using GloVe [92] on Wikipedia 2014 + Gigaword 5 with 100 dimensions.

3: https://nlp.stanford.edu/projects/glove/

In addition, we set a number of dimensions for both encoder and decoder hidden vectors to 200 and utilized a teacher forcing technique for efficient training where the teacher forcing ratio is set to 0.5 in our experiments.

5.3.3.3 Comparison of XERG and baselines

We compared our proposed learning framework, XERG, with baselines that are listed as follows:

Table 5.1: Results in terms of percentage (%) of each method for generating relationship explanations for entity sets.

Method	BLEU@1	BLEU@2	BLEU@3	BLEU@4	ROUGE-L	METEOR
RAND	25.28	13.57	10.01	8.41	**28.02**	17.19
ENT	26.21	14.14	10.38	8.63	27.21	17.04
XERG(comp)	**26.74**	**14.29**	**10.44**	**8.67**	27.38	17.36
XERG(inform)	26.44	14.18	10.40	8.64	27.26	**17.43**
XERG(comp, inform)	25.27	13.65	9.95	8.27	25.01	16.39

- **ENT**: the encoder-decoder network using only the entity information in the KG. Thus, the input path is represented as follows:

$$(e_1, < \text{pad} >, e_2, < \text{pad} >, \ldots, e_{|E|}) \tag{5.13}$$

  where e_i belongs to each entity set E.
- **RAND**: the encoder-decoder network using randomly selected paths for entity sets.

For XERG, we considered three different settings of $\alpha_r \in \{0.0, 1.0\}$, $\beta_r \in \{0.0, 1.0\}$ that control the degrees of the two criteria for rewards, *i.e.* r^{comp} and r^{inform}. For computing r^{inform}, we opted to use METEOR as it was demonstrated that this metric achieved the best correlation with human judgments among the other evaluation metrics including BLEU and ROUGE-L [93]. To carefully measure the informativeness of the paths, we excluded any entity names from the generated explanations before computing r^{inform} as we expect the ERR network to find the paths that could lead to the generation of many relevant vocabularies. We denote three variations of XERG as XERG(comp) for $\alpha_r = 1.0, \beta_r = 0.0$, XERG(inform) for $\alpha_r = 0.0, \beta_r = 1.0$, and XERG(comp, inform) for $\alpha_r = 1.0, \beta_r = 1.0$.

For every method, we trained them using the data with a batch size of 64 for a maximum number of 50 epochs. We adopted an early stopping criterion using METEOR on the validation set in which we set the number of patience to 10 epochs.

5.3.4 Results

5.3.4.1 RQ1 - *Do the information within the KG and its structure help generating better relationship explanations of entities?*

According to the performance of baselines and XERG with different settings in terms of BLEU@n (*i.e.* $n \in [1, 4]$), ROUGE-L, and METEOR shown in Table 5.1, XERG(comp) outperformed the baselines for most evaluation metrics including the highest BLEU@1 at 26.74, BLEU@2 at 14.29, BLEU@3 at 10.44 and BLEU@4 at 8.67, while XERG(inform) achieved the highest METEOR at 17.43 and its performance in terms of other metrics were comparable to those achieved by XERG(comp). Although, the highest ROUGE-L was achieved by the RAND baseline, its results of the other metrics were the lowest when comparing to the best two variations of XERG (*i.e.* XERG(comp) and XERG(inform)).

A Tukey's HSD test shows that the differences between our learning framework with three diferrent settings and RAND baseline in terms of

ROUGE-L are statistically significant with $p = 0.0010$. On the other hand, the test in terms of METEOR shows that ENT baseline is significantly different from XERG(comp) with $p = 0.0167$, XERG(inform) with $p = 0.0018$, and XERG(comp, inform) with $p = 0.0010$.

Figure 5.4 shows the actual path with the generated explanation of each entity set by the ENT baseline and our best proposed learning framework with two different settings. The examples of the generated relationship explanation revealed that utilizing the entity information as well as the predicate information within the KG through the entity relationship reasoning could result in a better relationship explanation for a set of entities.

5.3.4.2 RQ2 - *Which criterion for determining the rewards (r^{comp} and r^{inform}) for the entity relationship reasoning is more effective for generating the explanations?*

Refer to Figure 5.4, utilizing r^{comp} which controls the path completeness allows the reasoning network to find more complete paths that cover more entities in the entity set. However, the complete paths found by the reasoning network tend to be short and do not always include the predicates that indicate the information that is necessary for generating the relationship explanation for some cases. For instance, the first example path in Figure 5.4 by XERG(comp) which utilized a hundred percent of r^{comp} does not only contain the useful predicates such as *actor.film* and *award_category.nominees*, but also the predicates *book_subject.works* and *written_work.subjects* that are linked to the entity "Golden Globe Award" at the end. These two predicates do not necessarily interpret the relationship between the entity "Golden Globe Award" and other entities in the set, such that the learning framework could not infer the important information and not generate the explanation including the entity "Golden Globe Award".

On the other hand, utilizing r^{inform} which controls the informativeness of the paths could not support the reasoning network to find as many complete paths as r^{comp}, however, r^{inform} could suggest the reasoning network to find important predicates for the relationship explanation generation. Take the second actual result path by XERG(inform) as the example, although the path is far from complete as it does not contain every entity in a set, but rather contains multiple random entities, the path includes the predicate *football_player.position_s* which results in the generation to predict the vocabulary "played" in the context of "PERSON **played** SPORT" in the generated explanation as opposed to the ENT baseline which did not utilize any predicate information and, as a result, it could not predict the vocabulary "played" in the explanation.

To answer RQ2, both criteria for determining rewards, r^{comp} and r^{inform}, are relatively important as long as they could encourage the relationship reasoning network to discover informative paths including the important predicates.

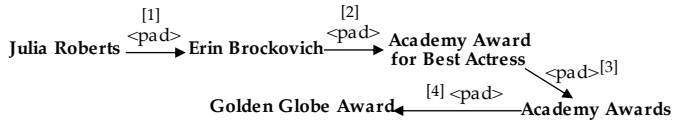
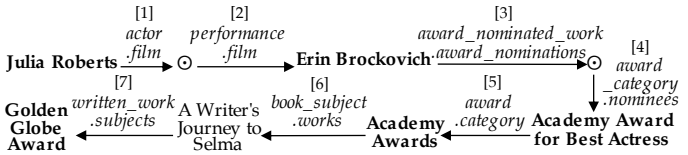
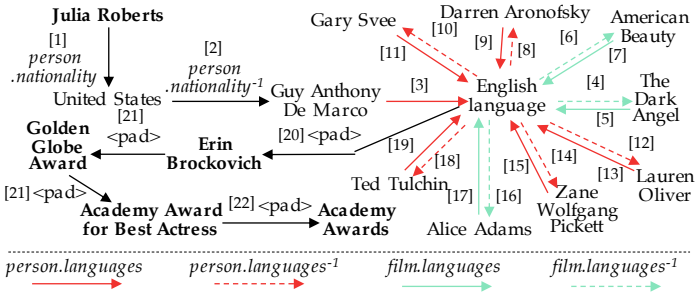
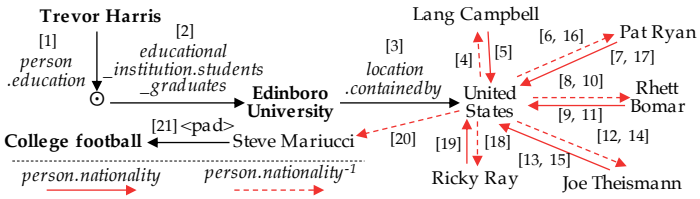
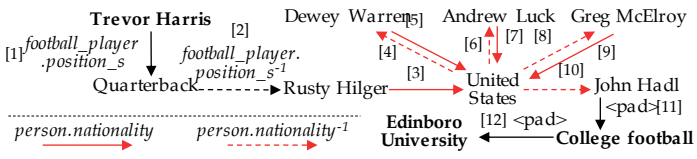
Entity Set and Ground Truth Explanation	Method	Path	Generated Explanation
{Julia Roberts, Erin Brockovich (film), Academy Award for Best Actress, Academy Awards, Golden Globe Award} - **Julia Roberts** has won three **Golden Globe Awards** nominations and has been nominated **Academy Awards** for her film acting winning the **Academy Award for Best Actress** for her performance in **Erin Brockovich**	ENT		**Julia Roberts** received a **Golden Globe Award** nomination for **Academy Award for Best Actress** for best for for best for for for for for for for for for for for for for best for her performance
	XERG (comp)		**Julia Roberts** received her nominations for **Academy Award for Best Actress** for best supporting actress for her performance for her performance in **Erin Brockovich**
	XERG (inform)		**Julia Roberts** received a **Golden Globe Award** nomination for best best for best for for for for for for for for for for for for for for for for for for for
{Trevor Harris, Edinboro University, College football} - **Trevor Harris** played **college football** at **Edinboro University**	ENT		**Trevor Harris** was **College football Edinboro University College football Edinboro University College football Edinboro University**
	XERG (comp)		**Trevor Harris** played **College football** in the **Edinboro University College football Edinboro University Edinboro University College football**
	XERG (inform)		**Trevor Harris** played **College football** at **Edinboro University**

Figure 5.4: The actual results including the path and generated explanation for the entity set by the ENT baseline and our learning framework with best two different settings (XERG(comp) and XERG(comp)). For paths, each number in square brackets attached to each predicate represents i-th of step for a predicate, and ⊙ represents a CVT (Compound Value Type) within Freebase for representing data referring to multiple fields. We represent the reverse edge of each predicate using r^{-1} and for simplicity, we omit any self-loops from the path.

5.4 Summary

We propose a learning framework, namely XERG, for generating a textual relationship explanation for a set of entities by fully utilizing the KG through the RL which allows us to discover informative paths with crucial information (*e.g.* relevant predicates and entities for each entity set) for generating descriptive relationship explanations. The experimental results on our new dataset suggested that our learning framework could generate a better relationship explanation for an entity set than baselines.

Conclusion 6

In this chapter, we conclude the three research proposals according to the goal of thesis as well as discuss the possible directions for future works.

6.1 Summary

The ultimate goal of this thesis is to support search users in searching for information regarding the entities. We addressed the limitation of the current entity-oriented search functionalities and raised the research problems that could alleviate those limitations in which we proposed the approaches that incorporate fine-grained entity information in terms of data representation and complexity of relationship within the KG to achieve the goal of each research proposal. The proposed research problems are listed as follows:

1. **Entity Ranking based on Queries with Modifiers**: In this study, we proposed two methods for ranking a set of entities based on a given search query by considering different types of modifiers (*i.e.* one corresponding to the qualitative properties and one corresponding to quantitative properties) so that we could provide a ranked listed of entities for a more number of search queries to the users. Our first proposed method is so called the Web-based entity ranking that involves measuring the relatedness of entities to a given query using the similarity scores between a query and entity type names and the frequency of entities in search results returned in response to the query. For the second method, we proposed the property-based entity ranking method which includes the part of identifying a property corresponding to each modifier in a query in which we proposed the property identification method based on the combination of seven proposed criteria including: 1) the frequency of property values including the modifier, 2) the co-occurrence of the modifier and property names on the Web, 3) the difference in property value distributions of entities in the search results, 4) the similarity between properties and modifiers, 5) the co-occurrence of major property value types and modifiers on the Web, 6) the difference between co-occurrence of property values and entity names on the Web, and 7) the frequency of properties including the modifier. Experimental results showed that our property identification method could identify more relevant properties, and the combination of our Web-based and property-based entity ranking methods could return more relevant entities at the top rank of the list.
2. **Identifying Entity Properties from Text with Zero-Shot Learning**: We focused on identifying properties of entities from text in this study as the identified properties can be used to support the

users with entity-oriented search in various ways. For example, we could construct a more relevant entity card using the relevant entity properties according to the content inside the documents clicked by the user or those at the top rank of the search results. To identify the entity properties, we proposed the neural network-based model to the problem of multi-label sentence classification that classifies sentences as representing properties given a target entity mentioned in the text. We considered the problem of ZSL that occurs when the sentences of certain properties are unavailable during training. To ease this problem, we incorporated embeddings of properties constructed by different techniques based on components of the KG as well as an entity type of the target entity to narrow down the potential properties represented in the sentences. There are five main components in our proposed neural network-based model including: 1) a word embedding that maps words in the given text to low-dimensional word vectors, 2) a bidirectional LSTM that generates a sentence-level representation of the text, 3) a mapping that learns the transformation between representations of sentences and properties, 4) an entity type that gives the emphasis on the properties that are likely to be represented by the sentence, and 5) a sigmoid function that produces probabilities over all properties. Experimental results showed that utilizing both the entity type and property embeddings improves the performance in identifying both seen and unseen properties.

3. **Explaining Relationship of Entity Sets**: For this research problem, we attempt to incorporate an explanation of relationship of entities on the entity-oriented search functionalities that involve multiple entities such as entity recommendation since the explanations could allow the users to understand how and why the entities are related and suggested in the first place. As a result, it could improve the user's engagement on such functionalities. To provide the relationship explanation of entities, we proposed a learning framework, namely XERG (*i.e.* eXplaining Entity Relationship with knowledge Graph), for generating a textual relationship explanation of a set of entities by fully utilizing the KG through the RL which allows us to discover informative paths with crucial information for generating descriptive relationship explanations. our XERG includes two main components (*i.e.* the Entity Relationship Reasoning (ERR) network and the encoder-decoder network). The ERR network learns to explore an informative path between entities in the given entity set, while the encoder-decoder network takes as an input the reasoning path and generates a textual relationship explanation as an output along with feedback that informs the ERR network about the informativeness of the path according to the generated explanation. By using our new dataset, the experimental results showed that our learning framework could generate a better relationship explanation for an entity set than baselines.

Finally, the technical and social contributions of this thesis are listed as follows:

- **Technical Contribution**
  - We introduced two entity ranking methods for queries including modifiers, *i.e.* the Web-based entity ranking and property-

based entity ranking, and proposed a property identification method to be used in the property-based entity ranking based on our proposed seven criteria that are suitable for different types of modifiers.
- We introduced a neural network-based model for multi-label sentence classification to identify properties represented in sentences, while also addressed the ZSL problem by extending the proposed model to utilize property embeddings constructed based on a knowledge graph embedding.
- We introduced a learning framework consisting of the Entity Relationship Reasoning and encoder-decoder networks for a generation of a textual relationship explanation for an entity set by fully utilizing the KG structure and its information.

► **Social Contribution**

- By supporting people with entity-oriented search, it allows them to access the information related to entities efficiently in a variety of areas such as health, business, government and education since it reduces the time it takes to locate desired information to accomplish their day to day tasks, and more importantly, it improves their learning rates as they can obtain new information easily.

6.2 Future Directions

Although the research proposals in this thesis focused on various approaches to mitigate the current limitations of the existing entity-oriented search functionalities, there are still many other problems regarding the search for entity-related information that could be explored and could be benefited from the fine-grained information of entities within the KG. Taken as the example the question-answering task, this task allows the search users to obtain the answer based on the question regarding the entities. However, there are some types of questions that remain unanswered as those questions demand the answers related to the numerical information such as “What is the average clock rate of the popular Intel processors?” or those that require the detailed analysis based on the complex information within the KG such as “Has any actors played the same character in as many films as Robert Downey Jr. as Tony Stark?”.

Bibliography

[1] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. 'Ad-hoc object retrieval in the web of data'. In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 771–780 (cited on pages 1, 28).

[2] Jiafeng Guo et al. 'Named entity recognition in query'. In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2009, pp. 267–274 (cited on pages 1, 13, 28).

[3] Xiaoxin Yin and Sarthak Shah. 'Building taxonomy of web search intents for name entity queries'. In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 1001–1010 (cited on pages 1, 13, 28).

[4] Rianne Kaptein et al. 'Entity ranking using Wikipedia as a pivot'. In: *CIKM*. 2010, pp. 69–78 (cited on pages 4, 9).

[5] Gianluca Demartini et al. 'Why finding entities in Wikipedia is difficult, sometimes'. In: *Information Retrieval* 13.5 (2010), pp. 534–567 (cited on pages 4, 9).

[6] Krisztian Balog, Marc Bron, and Maarten De Rijke. 'Query modeling for entity search based on terms, categories, and examples'. In: *ACM TOIS* 29.4 (2011), 22:1–22:31 (cited on pages 4, 10).

[7] Marc Bron, Krisztian Balog, and Maarten De Rijke. 'Ranking related entities: components and analyses'. In: *CIKM*. 2010, pp. 1079–1088 (cited on pages 4, 10).

[8] Hugo Zaragoza et al. 'Ranking very many typed entities on wikipedia'. In: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM. 2007, pp. 1015–1018 (cited on page 4).

[9] Jovan Pehcevski, Anne-Marie Vercoustre, and James A Thom. 'Exploiting locality of Wikipedia links in entity ranking'. In: *European Conference on Information Retrieval*. Springer. 2008, pp. 258–269 (cited on page 4).

[10] Mike Mintz et al. '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. 2009, pp. 1003–1011 (cited on pages 5, 11, 28).

[11] Sebastian Riedel, Limin Yao, and Andrew McCallum. 'Modeling relations and their mentions without labeled text'. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2010, pp. 148–163 (cited on pages 5, 11, 28, 37).

[12] Raphael Hoffmann et al. '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. 2011, pp. 541–550 (cited on pages 5, 11, 28).

[13] Mihai Surdeanu et al. '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. 2012, pp. 455–465 (cited on pages 5, 11, 28).

[14] Nikos Voskarides et al. 'Learning to explain entity relationships in knowledge graphs'. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, pp. 564–574 (cited on pages 5, 12, 45).

[15] Jizhou Huang et al. 'Learning to Explain Entity Relationships by Pairwise Ranking with Convolutional Neural Networks.' In: *IJCAI*. 2017, pp. 4018–4025 (cited on pages 5, 12, 45).

[16] Nikos Voskarides, Edgar Meij, and Maarten de Rijke. 'Generating descriptions of entity relationships'. In: *European Conference on Information Retrieval*. Springer. 2017, pp. 317–330 (cited on pages 5, 12, 45).

[17] Shady Elbassuoni and Roi Blanco. 'Keyword search over RDF graphs'. In: *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM. 2011, pp. 237–242 (cited on pages 9, 23).

[18] Vagelis Hristidis and Yannis Papakonstantinou. 'Discover: Keyword search in relational databases'. In: *VLDB*. 2002, pp. 670–681 (cited on page 9).

[19] Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das. 'DBXplorer: A system for keyword-based search over relational databases'. In: *ICDE*. 2002, pp. 5–16 (cited on page 9).

[20] B Aditya et al. 'Banks: Browsing and keyword searching in relational databases'. In: VLDB Endowment. 2002, pp. 1083–1086 (cited on page 9).

[21] Lawrence Page et al. *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab, 1999 (cited on page 9).

[22] Lei Zou et al. 'Natural language question answering over RDF: a graph data driven approach'. In: *SIGMOD*. 2014, pp. 313–324 (cited on page 9).

[23] Christina Unger et al. 'Template-based question answering over RDF data'. In: *WWW*. 2012, pp. 639–648 (cited on page 9).

[24] Daniel Gerber and A-C Ngonga Ngomo. 'Bootstrapping the linked data web'. In: *1st Workshop on Web Scale Knowledge Extraction@ ISWC*. Vol. 2011. 2011 (cited on page 9).

[25] Arjen P De Vries et al. 'Overview of the INEX 2007 entity ranking track'. In: *INEX*. 2007, pp. 245–251 (cited on page 9).

[26] Gianluca Demartini et al. 'Overview of the INEX 2008 entity ranking track'. In: *INEX*. 2008, pp. 243–252 (cited on page 9).

[27] Gianluca Demartini, Tereza Iofciu, and Arjen P De Vries. 'Overview of the INEX 2009 entity ranking track'. In: *INEX*. 2009, pp. 254–264 (cited on page 9).

[28] Krisztian Balog, Pavel Serdyukov, and Arjen P de Vries. 'Overview of the TREC 2010 entity track'. In: *TREC*. 2010 (cited on page 9).

[29] Lanbo Zhang, Yi Zhang, and Yunfei Chen. 'Summarizing highly structured documents for effective search interaction'. In: *SIGIR*. 2012, pp. 145–154 (cited on page 10).

[30] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 'Kernel methods for relation extraction'. In: *Journal of machine learning research* 3.Feb (2003), pp. 1083–1106 (cited on page 11).

[31] Aron Culotta and Jeffrey Sorensen. 'Dependency tree kernels for relation extraction'. In: *Proceedings of the 42nd annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2004, p. 423 (cited on page 11).

[32] Razvan C Bunescu and Raymond J Mooney. '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. 2005, pp. 724–731 (cited on page 11).

[33] Zhou GuoDong et al. 'Exploring various knowledge in relation extraction'. In: *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2005, pp. 427–434 (cited on page 11).

[34] Raymond J Mooney and Razvan C Bunescu. 'Subsequence kernels for relation extraction'. In: *Advances in neural information processing systems*. 2006, pp. 171–178 (cited on page 11).

[35] Razvan Bunescu and Raymond Mooney. 'Learning to extract relations from the web using minimal supervision'. In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. 2007, pp. 576–583 (cited on page 11).

[36] Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. '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. 2012, pp. 721–729 (cited on page 11).

[37] Wei Xu et al. '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. 2013, pp. 665–670 (cited on page 11).

[38] Bonan Min et al. '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*. 2013, pp. 777–782 (cited on page 11).

[39] Alan Ritter, Luke Zettlemoyer, Oren Etzioni, et al. 'Modeling missing data in distant supervision for information extraction'. In: *Transactions of the Association for Computational Linguistics* 1 (2013), pp. 367–378 (cited on page 11).

[40] Richard Socher et al. '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. 2012, pp. 1201–1211 (cited on pages 11, 40).

[41] Daojian Zeng et al. 'Relation classification via convolutional deep neural network'. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 2014, pp. 2335–2344 (cited on pages 11, 40).

[42] Dongxu Zhang and Dong Wang. 'Relation classification via recurrent neural network'. In: *arXiv preprint arXiv:1508.01006* (2015) (cited on pages 11, 40).

[43] Yan Xu et al. '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*. 2015, pp. 1785–1794 (cited on page 11).

[44] Peng Zhou et al. '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. 2016, pp. 207–212 (cited on page 11).

[45] Daojian Zeng et al. 'Distant supervision for relation extraction via piecewise convolutional neural networks'. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 1753–1762 (cited on page 11).

[46] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 'Attribute-based classification for zero-shot visual object categorization'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.3 (2014), pp. 453–465 (cited on page 11).

[47] Richard Socher et al. 'Zero-shot learning through cross-modal transfer'. In: *Advances in neural information processing systems*. 2013, pp. 935–943 (cited on page 11).

[48] Andrea Frome et al. 'Devise: A deep visual-semantic embedding model'. In: *Advances in neural information processing systems*. 2013, pp. 2121–2129 (cited on page 11).

[49] Yongqin Xian et al. 'Latent embeddings for zero-shot classification'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 69–77 (cited on page 11).

[50] Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, et al. 'Predicting deep zero-shot convolutional neural networks using textual descriptions'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4247–4255 (cited on page 12).

[51] Scott Reed et al. 'Learning deep representations of fine-grained visual descriptions'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 49–58 (cited on page 12).

[52] Antoine Bordes et al. 'Translating embeddings for modeling multi-relational data'. In: *Advances in neural information processing systems*. 2013, pp. 2787–2795 (cited on pages 12, 29, 34, 37, 39, 40, 53).

[53] Rik Koncel-Kedziorski et al. 'Text Generation from Knowledge Graphs with Graph Transformers'. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 2284–2293 (cited on page 12).

[54] Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 'Graph-to-Sequence Learning using Gated Graph Neural Networks'. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 273–283 (cited on page 12).

[55] Linfeng Song et al. 'A Graph-to-Sequence Model for AMR-to-Text Generation'. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 1616–1626 (cited on page 12).

[56] Diego Marcheggiani and Laura Perez-Beltrachini. 'Deep Graph Convolutional Encoders for Structured Data to Text Generation'. In: *Proceedings of the 11th International Conference on Natural Language Generation*. 2018, pp. 1–9 (cited on page 12).

[57] Christos Faloutsos, Kevin S McCurley, and Andrew Tomkins. 'Fast discovery of connection subgraphs'. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004, pp. 118–127 (cited on page 12).

[58] Hanghang Tong and Christos Faloutsos. 'Center-piece subgraphs: problem definition and fast solutions'. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2006, pp. 404–413 (cited on page 12).

[59] Gjergji Kasneci, Shady Elbassuoni, and Gerhard Weikum. 'Ming: mining informative entity relationship subgraphs'. In: *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM. 2009, pp. 1653–1656 (cited on page 12).

[60] Lujun Fang et al. 'Rex: explaining relationships between entity pairs'. In: *Proceedings of the VLDB Endowment* 5.3 (2011), pp. 241–252 (cited on page 12).

[61] Xinpeng Zhang, Yasuhito Asano, and Masatoshi Yoshikawa. 'A generalized flow-based method for analysis of implicit relationships on wikipedia'. In: *IEEE Transactions on Knowledge and Data Engineering* 25.2 (2011), pp. 246–259 (cited on page 12).

[62] Giuseppe Pirrò. 'Explaining and suggesting relatedness in knowledge graphs'. In: *International Semantic Web Conference*. Springer. 2015, pp. 622–639 (cited on page 12).

[63] Stephan Seufert et al. 'Espresso: explaining relationships between entity sets'. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM. 2016, pp. 1311–1320 (cited on page 12).

[64] Stephan Seufert et al. 'Instant Espresso: Interactive Analysis of Relationships in Knowledge Graphs'. In: *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee. 2016, pp. 251–254 (cited on page 12).

[65] Giuseppe Pirrò and Alfredo Cuzzocrea. 'RECAP: building relatedness explanations on the web'. In: *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee. 2016, pp. 235–238 (cited on page 12).

[66] Richard Socher et al. 'Reasoning with neural tensor networks for knowledge base completion'. In: *Advances in neural information processing systems*. 2013, pp. 926–934 (cited on page 12).

[67] Zhen Wang et al. 'Knowledge graph embedding by translating on hyperplanes'. In: *Twenty-Eighth AAAI conference on artificial intelligence*. 2014 (cited on page 12).

[68] Yankai Lin et al. 'Learning entity and relation embeddings for knowledge graph completion'. In: *Twenty-ninth AAAI conference on artificial intelligence*. 2015 (cited on page 12).

[69] Ni Lao, Tom Mitchell, and William W Cohen. 'Random walk inference and learning in a large scale knowledge base'. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2011, pp. 529–539 (cited on page 12).

[70] Ni Lao and William W Cohen. 'Relational retrieval using a combination of path-constrained random walks'. In: *Machine learning* 81.1 (2010), pp. 53–67 (cited on page 12).

[71] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 'Compositional Vector Space Models for Knowledge Base Completion'. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, pp. 156–166 (cited on page 12).

[72] Kristina Toutanova et al. 'Compositional learning of embeddings for relation paths in knowledge base and text'. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 1434–1444 (cited on page 12).

[73] Wenhan Xiong, Thien Hoang, and William Yang Wang. 'Deeppath: A reinforcement learning method for knowledge graph reasoning'. In: *arXiv preprint arXiv:1707.06690* (2017) (cited on page 12).

[74] Nina Phan, Peter Bailey, and Ross Wilkinson. 'Understanding the relationship of information need specificity to search query length'. In: *SIGIR*. 2007, pp. 709–710 (cited on page 13).

[75] Tessa Lau and Eric Horvitz. 'Patterns of search: analyzing and modeling web query refinement'. In: *UM99 User Modeling*. Springer, 1999, pp. 119–128 (cited on page 13).

[76] Hongbo Deng, Michael R Lyu, and Irwin King. 'A generalized Co-HITS algorithm and its application to bipartite graphs'. In: *KDD*. 2009, pp. 239–248 (cited on page 15).

[77] Tomas Mikolov et al. 'Distributed representations of words and phrases and their compositionality'. In: *Advances in neural information processing systems*. 2013, pp. 3111–3119 (cited on pages 18, 39).

[78] Joseph L Fleiss. 'Measuring nominal scale agreement among many raters.' In: *Psychological bulletin* 76.5 (1971), p. 378 (cited on page 23).

[79] Kalervo Järvelin and Jaana Kekäläinen. 'Cumulated gain-based evaluation of IR techniques'. In: *ACM Transactions on Information Systems (TOIS)* 20.4 (2002), pp. 422–446 (cited on page 25).

[80] Horatiu Bota, Ke Zhou, and Joemon M Jose. '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. 2016, pp. 131–140 (cited on page 28).

[81] Sepp Hochreiter and Jürgen Schmidhuber. 'Long short-term memory'. In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cited on page 31).

[82] Mike Schuster and Kuldip K Paliwal. 'Bidirectional recurrent neural networks'. In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681 (cited on page 32).

[83] Diederik P Kingma and Jimmy Ba. 'Adam: A method for stochastic optimization'. In: *arXiv preprint arXiv:1412.6980* (2014) (cited on pages 34, 49, 52).

[84] Justus J Randolph. 'Free-Marginal Multirater Kappa (multirater K [free]): An Alternative to Fleiss' Fixed-Marginal Multirater Kappa.' In: *Online submission* (2005) (cited on page 38).

[85] Laurens van der Maaten and Geoffrey Hinton. 'Visualizing data using t-SNE'. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605 (cited on page 43).

[86] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (cited on pages 47, 48).

[87] Kyunghyun Cho et al. 'Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation'. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1724–1734 (cited on page 50).

[88] Fei Wu and Daniel S Weld. 'Open information extraction using Wikipedia'. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics. 2010, pp. 118–127 (cited on page 52).

[89] Kishore Papineni et al. 'BLEU: a method for automatic evaluation of machine translation'. In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2002, pp. 311–318 (cited on page 53).

[90] Chin-Yew Lin. 'ROUGE: A Package for Automatic Evaluation of Summaries'. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, 2004, pp. 74–81 (cited on page 53).

[91] Satanjeev Banerjee and Alon Lavie. 'METEOR: An automatic metric for MT evaluation with improved correlation with human judgments'. In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 2005, pp. 65–72 (cited on page 53).

[92] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 'Glove: Global vectors for word representation'. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543 (cited on page 53).

[93] Peter Anderson et al. 'Spice: Semantic propositional image caption evaluation'. In: *European Conference on Computer Vision*. Springer. 2016, pp. 382–398 (cited on page 54).

Publications

Academic Journal

1. Wiradee Imrattanatrai，Makoto P. Kato, and Masatoshi Yoshikawa, ゼロショット学習によるテキストからのエンティティプロパティ同定．日本データベース学会和文論文誌，Vol.18-J，Article No.10，2020.3
2. Wiradee Imrattanatrai, Makoto P. Kato, Katsumi Tanaka, and Masatoshi Yoshikawa, Entity Ranking for Queries with Modifiers Based on Knowledge Bases and Web Search Results, In *IEICE Transactions on Information and Systems* , Vol. E101.D, Issue 9, pp 2279-2290, 2018.

International Conference

1. Wiradee Imrattanatrai, Makoto P. Kato, and Masatoshi Yoshikawa, Explaining Relationship of Entity Sets with Knowledge Graph, *The 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, **[Under Reviewed]**
2. Wiradee Imrattanatrai, Makoto P. Kato, and Masatoshi Yoshikawa, Identifying Entity Properties from Text with Zero-Shot Learning, In *Proceeding of the 42nd International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR 2019)*, pp 195-204, 2019.
3. Wiradee Imrattanatrai, Makoto P. Kato, and Katsumi Tanaka, Entity Search by Leveraging Attributive Terms in Sentential Queries over RDF Data, In *Proceeding of the 16th IEEE/WIC/ACM International Conference on Web Intelligence (WI 2017)*, pp 769-776, 2017.

Domestic Symposium

1. Wiradee Imrattanatrai, Makoto P. Kato, and Masatoshi Yoshikawa, ゼロショット学習によるテキストからのエンティティプロパティ同定, 第11回データ工学と情報マネジメントに関するフォーラム（DEIM 2019), 長崎市 , 2019年3月.
2. Wiradee Imrattanatrai, Makoto P. Kato, and Katsumi Tanaka, Entity Search by Leveraging Attributive Terms in Sentential Queries over RDF Data, 第9回データ工学と情報マネジメントに関するフォーラム （DEIM 2017), 高山市 , 2017年2月.

Workshop

1. Wiradee Imrattanatrai, Search & Summarization of Entity-related Information, SIGIR Tokyo-Beijing Joint Workshop 2019, 北京, 2019年 1月.
2. Wiradee Imrattanatrai, Entity Search by Leveraging Attributive Terms in Sentential Queries over RDF Data, iDB特別セッション, Webとデータベースに関するフォーラム (WebDB Forum 2017), 東京都文京区, 2017年9月.
3. Wiradee Imrattanatrai, Instance Suggestion based on Knowledge Base and Web Search Results, 関西データベースワークショップ2016, 神戸市, 2016年9月.