EXPLOITING KNOWLEDGE GRAPHS FOR ENTITY-CENTRIC PREDICTION

BY

SHAN JIANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Doctoral Committee:

Professor Chengxiang Zhai, Chair
Professor Jiawei Han
Assistant Professor Jian Peng
Associate Professor Qiaozhu Mei, University of Michigan

ABSTRACT

As a special kind of "big data", text data can be regarded as data reported by human sensors. Since humans are far more intelligent than physical sensors, text data contains directly useful information and knowledge about the real world, making it possible to make predictions about real-world phenomena based on text. As all application domains involve humans, text-based prediction has widespread applications, especially for optimization of decision making.

While the problem of text-based prediction resembles text classification when formulated as a supervised learning problem, it is more challenging because the variable to be predicted is generally not directly mentioned in the text data and thus there is a "semantic gap" between the target variable and the surface features that are often used for representing text data in conventional approaches. How to bridge such a gap is a key technical challenge, but has not been well studied in the existing work. In this thesis, we propose to leverage the increasingly available knowledge graphs on the Web to bridge this gap. We propose to bridge this gap by using knowledge graph to make text representation more focused on elements in a knowledge graph that are relevant to the prediction task. We mainly focus on two a family of text-based prediction – entity-centric classification and regression where the response variable can be treated as an attribute of a group of central entities.

As a form of knowledge representation, knowledge graphs have widespread applications in information retrieval, text mining, and natural language processing. Many knowledge graphs have been constructed and applied to diverse, real-world applications. The knowledge graph can help to enhance the interpretability of the textual information from the perspective of predictive analytics, and hence discovers more effective features. Despite the great success made in the application of knowledge graph in various domains, one of the main deficiencies of many existing works is that the knowledge graph applied in the application is pre-constructed, which remains unchanged when applied to very different specific application tasks. Such a static task-independent knowledge graph, while useful, is non-optimal for any specific application due to the unnecessary cost from processing large amounts of non-relevant knowledge as well as the insufficient coverage of task-specific knowledge.

To address this limitation, we propose to construct a task-aware knowledge graph (TAKG) which would only contain the relevant knowledge to a particular task and absorb additional relevant knowledge from the data used in a particular task. We present a general formal framework for constructing a task-aware knowledge graph, develop specific algorithms for

constructing a task-aware knowledge graph for entity-centric prediction in both knowledge-based and task-dependent ways, and apply it to a movie review categorization task.

We propose two methods to expand the knowledge graph. One is to discover new entities and relations by a jointly embedding model which learns embedding vector for each entity and relation. In this way, the specific relationships in a finer-granularity that is pre-defined by the knowledge graph can be identified between related entities. An alternative way is to use more general word relations, e.g., paradigmatic and syntagmatic relation to expand the knowledge graph by including loosely related entities. Both methods work under certain circumstances, but the former one is helpful in a wider range of applications.

We also make a systematic study of knowledge graph assisted feature engineering. We propose several different ways to construct knowledge graph-based features and investigate their performance in multiple real applications. Our study shows that different types of application may favor different ways of constructing knowledge graph-based features. We find that the coverage of the knowledge graph is important. If it cannot provide sufficient background knowledge, the effectiveness of the knowledge graph-based features will be impacted. Besides, the generated knowledge graph-based features can sometimes be very noisy, especially when the correlation between text and the response variables are weak. To distinguish the signal features from the noise, we propose a two-stage filtering method to further prune the features. Our experiment result shows that the pruned knowledge graph-based features have strong predictive power, which again confirms that leveraging text data is promising for real-world phenomenon prediction.

To Yucheng; to my parents

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Chengxiang Zhai. Cheng is the best advisor who is always supportive and friendly to his students. He is one of the most considerate and kind-hearted people I've ever met. Without Cheng's continuous guidance, encouragement and help, this thesis wouldn't be completed. I'm deeply influenced by Cheng's broad vision of research, modest personality, and serious attitude to work.

I would like to thank my committee members: Professor Jiawei Han, Professor Jian Peng and Professor Qiaozhu Mei, for their great feedback and suggestion on this thesis. I'm appreciative of Professor Han and Professor Zhai's generous help for me to get through all the trouble when scheduling the defense. Professor Han has offered many valuable suggestions and feedbacks on the thesis. Taking two semesters of Professor Han's courses stimulated my interest in data mining. Professor Peng has given me insightful comments and enlightening discussion on a wide range of related topics. I'm deeply grateful to Professor Mei. I worked with Qiaozhu for more than two years on a text mining project. I have learned so much from Qiaozhu. I would like to express my appreciation not only for his technical guidance, but also for his continuous encouragement and support for my study and career choice. I am deeply indebted to both Cheng and Qiaozhu for helping me so much on developing the idea and polishing the paper. This thesis would not be possible without their support and help.

I have had great opportunities to work with many talented researchers in the past summer internship. I sincerely thank Yuening Hu, Changsung Kang, Yi Chang, Yi Liu for being such wonderful mentors.

I'm thankful to my colleagues in the TIMAN Group for their help. I also wish to thank my friends, Jingjing, Huan and Yujing. Thank them for the unforgettable moments spent together. I will treasure these memories forever in my life. I always received encouragement and power from my family and friends when I had a tough time during my Ph.D. study.

Finally, I would like to thank my dear husband Yucheng and my parents, for their love, and everything they've done for me. This thesis is dedicated to them.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Text data, broadly including all kinds of natural language text produced by humans (e.g., web pages, social media, email messages, news articles, government documents, and scientific literature), have been growing dramatically recently. As a special kind of big data, text data can be regarded as data generated by "human sensors" about our world, which is often much more informative and useful compared to the data generated by physical sensors due to the fact that humans are far more intelligent. This creates great opportunities for applying computational methods to mine big text data to discover all kinds of useful knowledge and support many data analytics applications, such as intelligent information system to accelerate biomedical discovery, improve homeland security, enable social media monitoring, and assist decision making in general.

Among many applications of text data mining, prediction of interesting variables based on relevant text data is especially interesting because it allows decision makers to "see" patterns behind data that they would otherwise not be able to see or easily see, thus leading to better decision making. For example, while non-text data may (indirectly) suggest a particular trend that consumers tend to like a particular feature of a product, product reviews may directly report such preferences by consumers. Since humans are involved in every application domain, text-based prediction has applications in virtually all the application domains. By exploiting the power of text information, it can help a variety of downstreaming applications, such as election forecast [1, 2], prediction of produce sales [3, 4], box office revenue [5].

While the formulation of a text-based prediction problem resembles a normal text classification problem, especially when we formulate both as a supervised learning problem, text-based prediction is generally much more challenging than a normal text classification task such as topic categorization or sentiment analysis because the attribute value to be predicted is in general not directly derivable from the text data and the correlation between the text input and the response variable can be very weak. For example, in the task of movie revenue prediction, text information such as movie reviews only delivers week signals. It is hard for both machine and human being to directly infer the revenue performance by reading the reviews. Previous studies showed that sentiment analysis is helpful for the task, but revenue performance is not only determined by people's opinion towards it. Another example is to predict election results, sentiment is also important, but it is not sufficient to only use the sentiment polarities to make accurate prediction. We need to dig into more details from the text information. The challenge is that raw text is often too noisy. Sometimes

a large part of a document is not relevant at all, and only a few snippets are relevant to the response variables we want to infer. The response variables are not directly mentioned or easily derivable from the text in many cases. That is, there is often a semantic gap between the attribute value to be predicted and the surface lexical features we can extract from the text data. How to bridge such a gap is a key technical challenge, but has not been well studied in the literature. In this thesis, we propose to leverage the increasingly available knowledge graphs on the Web to bridge this gap.

To understand why knowledge graph can help bridge the gap, we can analyze the challenge in text-based prediction of attribute values of entities from two perspectives – feature generation and feature generalization.

For feature generation, the most popular method is to use text-level surface lexical features (e.g., words or phrases). However, one problem with such surface lexical features is that the free-form of text contains much redundancy or noise, and lacks semantic discrimination. Deeper semantic feature representation of documents is needed to enable interpretation the text information for prediction purpose. For instance, if we want to classify online movie reviews collected from social media according to the movies that are being reviewed, we can find that it is easy to infer the movie label when people are talking about the directors or actors/actresses. Such kind of background knowledge can help a lot to identify the movie labels, but may not be easily deducible by only using surface lexical features. Another example of the semantic gap can be shown in movie revenue prediction task, which is to predict the box office revenue of movies. Discussion about the performance of actor/actress, impression of the director, and fondness of the story are all informative signals for the prediction task that can be discovered from the text data. Such a finer-grained analysis, however, would require additional knowledge on a deeper semantic layer (e.g., recognition of directors, actors/actresses and authors) which can not be fully satisfied by using word-level or phrase-level features. The main motivation for our work is to use a knowledge graph to bridge such a gap.

For feature generalization, a commonly-used type of features that are applicable to multiple applications is the intermediate features extracted from the text data. For example, sentiment polarity is a widely-used feature in many applications such as movie revenue prediction, election forecast. Though such intermediate features are extracted from text automatically and shown to be effective to a variety of applications, they are not universally applicable to every text-based prediction task. Intermediate features usually rely on some pre-set assumptions. Once the pre-set assumptions do not hold any more, we need to create new intermediate features. Since it is infeasible to exhaust all possibly useful intermediate features, constructing intermediate features is not an optimal solution to represent the text

data for making accurate prediction in all scenarios. To construct features that are not only effective to facilitate precise prediction but also interpretable and generalizable, we again propose to exploit knowledge graph. By identifying what kind of information is relevant to the task based on the background knowledge acquired from a knowledge graph, we can extract more relevant semantic content from the text, and consequently better capture the correlation between the text and the target variable that is to be predicted. The constructed features are derived from human knowledge provided by the knowledge graph, they are interpretable and can help to explore patterns behind the text data from the perspective of predicting real-world phenomena. Besides, the knowledge graph-based features do not depend on pre-set assumptions, thus are more generalizable for various applications in diverse domains.

Among all kinds of applications of text-based prediction, a large number of them are entity-based where the variables to be predicted are the attribute values of a set of entities. We refer to this type of prediction problem as *entity-centric prediction*, and the involved entities are called *central entities*. For example, predicting the rating of restaurants based on socially-generated comments, predicting the sales performance of products from consumer feedback, and predicting the revenue of movies based on reviews are all examples of entity-centric prediction. In this thesis, we mainly focus on entity-centric prediction, where knowledge graph can serve as a natural bridge to connect the textual information and the target variables. Target variables are associated with a group of central entities. Even though it is not clear how the target variables are related to the textual information, we can assume that information related to the central entities is latently related to the target variables in a certain way. We can easily extract information related to the central entities from the background text with the help of knowledge graph. In this way, the correlation between text and response can be better captured.

We propose to generate knowledge graph-based features to represent text data for entity-centric prediction tasks, and how to obtain the best-fit knowledge graph for a particular task is also an essential problem. Constructing a large knowledge graph has been traditionally labor-intensive. With the arrival of information age, a large amount of digital text is easily available on the Internet, which makes it possible to construct a very large knowledge graph automatically or semi-automatically by using information extraction techniques. As a result, a large number of knowledge graphs have been created and widely used in many text-related applications, such as WordNet [6], DBpedia [7], Freebase [8], YAGO [9], NELL [10], PROSPERA [11], DeepDive [12] and Knowledge Vault [13]. The growth of knowledge graphs available in the public domain makes such an approach especially appealing since as we have more knowledge graphs, our approach would also potentially become even more powerful.

There are extensive studies on applying knowledge graph in text-based prediction tasks and demonstrating the value of knowledge graph in those real applications [1], [2], [3], [5], [4]. In spite of great success achieved so far, the power of knowledge graph is still not fully exploited. One of the main reasons is that the construction and the application of knowledge graph are disjoint from each other – once a knowledge graph has been built, it tends to remain unchanged during its use in many different applications and cannot assimilate new useful knowledge from text data used in a particular application. As shown in Figure 1.1a, in the existing applications, knowledge graph is typically used as a knowledge resource fed into the application, but receives no feedback from it.

We argue that such a feedback process is generally necessary because a pre-constructed generic knowledge graph, no matter how large it is, would unlikely be able to fully cover all the knowledge needed in all kinds of application tasks; indeed, it is impossible to build a knowledge graph that encompasses the entire knowledge of human language even though theoretically such a "universal" knowledge graph exists. Another problem with directly using a pre-constructed generic knowledge graph without adaptation is that the knowledge graph contains large amounts of non-relevant knowledge to a particular task, which not only causes unnecessary computation, but also can be even "distracting" for a particular task (e.g., causing overfitting when used in a supervised learning program).

To incorporate such task-based feedback and prune any non-relevant knowledge in a generic knowledge graph, we propose to construct a *task-aware knowledge graph* (TAKG) which would only contain the relevant knowledge to a particular task and assimilate additional relevant knowledge from the data used in a particular task. In this way, we can build a knowledge graph that is customized to the specific entity-centric prediction problem, which not only provides the best support to the task but is also potentially helpful for similar tasks in the same domain.

The construction of knowledge graph can be considered as a process of encoding useful information from text into a graph structure which determines the semantics or domain-specific roles of words or phrases, while the application of knowledge graph in text-related applications is more like a reverse operation which decodes information embedded in the text with the help of a knowledge graph. The task-aware knowledge graph we propose in this thesis bridges the gap in the existing work between the construction and application of knowledge graph to allow for a potentially iterative process for constructing an increasingly relevant and complete knowledge graph customized toward optimizing the performance of a specific task. As shown in Figure 1.1b, feedback from the task is utilized to improve the knowledge graph in the construction of task-aware knowledge graph, and the knowledge graph and the task can mutually benefit from each other.

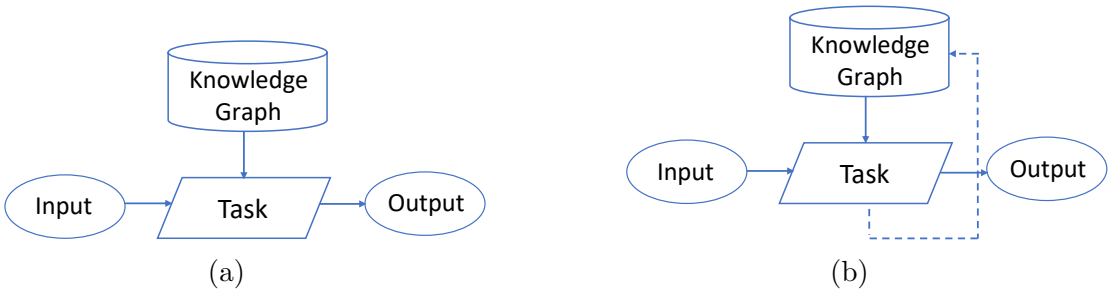


Figure 1.1: (a) Common way of applying knowledge graph. (b) Construction and application of task-aware knowledge graph

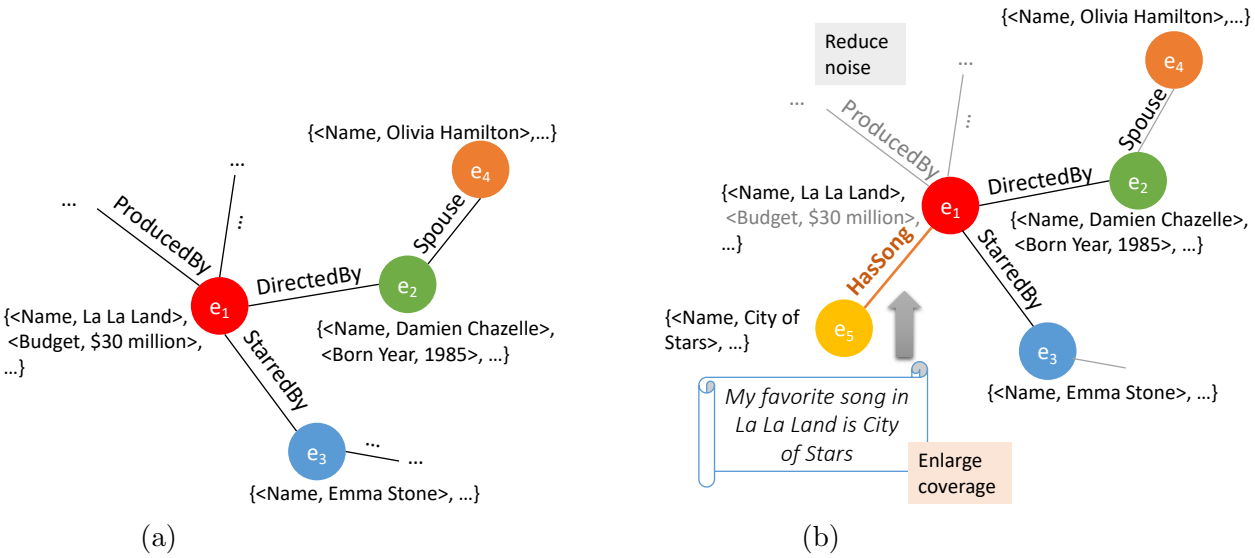


Figure 1.2: (a) An example of generic knowledge graph. (b) An example of how a task-aware knowledge graph is constructed

As an example, consider the task to annotate movie reviews with the movies being reviewed, which is a text classification task with movies as class labels. Part of a generic knowledge graph that can be used to solve this problem is shown in Figure 1.2a which includes movie entities and their relations with other kinds of entities. If we find that relations such as "*ProducedBy*" and attributes of movie entities such as "*Budget*" are not very useful or even bring in noise sometimes for the task, we can remove them from the knowledge graph. By trimming the knowledge graph, we obtain a more relevant and less noisy task-aware knowledge graph customized for our task. Besides, we can also discover new knowledge from the text data in our task (i.e., reviews), such as extracting new entities and relations, finding more attributes of entities. For example, we find a review "City of Stars is my favorite song in La La Land", indicating that "City of Stars" is a song of "La La Land" which is unknown to the knowledge graph. Then we can expand the task-aware

knowledge graph with this new knowledge. Such an expanded task-aware knowledge graph would then enable us to use "City of Stars" as a clue to annotate the review where this song is mentioned with the correct label "La La Land."

This example shows that in general the construction of a task-aware knowledge graph would include (1) trimming a large generic knowledge graph (to remove noise and increase efficiency) and (2) expanding it with any new knowledge that can be extracted from the text data used in a particular task. The process can also be repeated multiple times to iteratively adapting the knowledge graph to optimize the performance of a task.

In this thesis, we provide a formal definition of task-aware knowledge graph and propose a general formal framework for constructing a task-aware knowledge graph that supports both trimming and expanding a knowledge graph. A knowledge graph is composed of a set of entities and relations between them. The basic structure of a task-aware knowledge graph is similar, but it differs from a generic knowledge graph in that a task-aware knowledge graph also carries information about the relevance or usefulness of an entity or relation from the perspective of a particular task. The framework thus includes a task awareness measurement function, which measures the relevance of entities and relations to a task and guides the construction algorithm to find the precisely-targeted knowledge needed in a task.

While customized generation of knowledge graph is in general helpful for any applications, as an initial study of task-aware knowledge graph, we are particularly interested in those application tasks involving entity-centric prediction. The benefit of a knowledge graph for entity-centric prediction task is easy to obtain and construction of a task-aware knowledge graph for such tasks is more tractable. The constructed task-aware knowledge graph provides the better assistance to the specific entity-centric prediction task compared to a generic knowledge graph or a domain knowledge graph, as it is enhanced by learning new knowledge from its application in the task.

With entity-centric text prediction tasks in mind, we systematically study effectiveness of task-aware knowledge graph for such tasks. We follow the proposed general framework and develop new algorithms for constructing a task-aware knowledge graph for entity-centric prediction in both knowledge-based and task-dependent ways. The construction of task-aware knowledge graph can start from either an existing generic knowledge graph or a pre-generated seed knowledge graph which mainly contains the central entities. Trimming of the knowledge graph is done based on the task awareness measurement. The task-aware knowledge graph would be expanded if the coverage of the knowledge is not large enough and new relevant knowledge is available in the background text data. The expansion is data-dependent and only focuses on entities or relations with high task awareness, thus assimilating the relevant knowledge very efficiently and avoiding the completion of the whole

generic knowledge graph. The trimming and expansion can be carried out in turn repeatedly until the coverage is sufficient and the noise is ignorable.

When we get a task-aware knowledge graph specifically constructed for an entity-centric prediction task, the next question is how such a knowledge graph can help to optimize the performance in the task. This question is twofold: whether a knowledge graph (including a task-aware knowledge graph) is helpful at all; if so, how to leverage the knowledge graph to benefit the task.

To answer these two questions, we first study how a knowledge graph can be used to construct features for two representative tasks, classification and regression, and show that whether the idea of using knowledge graph is appealing. Our study shows that a generic knowledge graph is beneficial in general for some easier tasks like entity-centric classification. But it doesn't always help, especially when the task is very difficult and the gap is large. One limitation of the generic knowledge graph we observed in these experiments is that the knowledge graph does not provide sufficient background knowledge needed in the task. To see whether the extension of coverage is important, which is one of our main motivations to construct a task-aware knowledge graph, we propose an efficient algorithm to discover both paradigmatic and syntagmatic relations from the text corpora and use both relations to expand the knowledge graph. These are two fundamental relations between words based on their co-occurrence. Different from the well-organized relations provided by a knowledge graph, paradigmatic and syntagmatic relation can tell user some hint about there may exist a certain type of "vague" connection between two words. Even though they are not categorized into the pre-defined relation types, they can still help to find more relevant entities. We find that when the coverage is enlarged, even though by more general relationships rather than the pre-defined finer-granularity relationships in a knowledge graph, the performance the performance is significantly improved, indicating that constructing a customized knowledge graph with sufficient coverage is necessary. Paradigmatic and syntagmatic relations can work very well to help us to find good features in general. However, they are not powerful enough when more accurate relations in finer granularity are needed, and a task-aware knowledge graph which can discover more accurate relations would be a better choice in this case.

To find a better way to leverage knowledge graph in entity-centric prediction tasks, we systematically study knowledge graph-based feature construction for entity-centric prediction applications. We propose several ways to construct the knowledge graph-based features and compare their performance in different kinds of real-world applications. We find that entity-based features can work very well in some cases. However, when the semantic gap between the text and the response is larger, the problems become much more challenging and additional information from the local structure of the knowledge graph is needed. For

example, to exploit text data in more challenging tasks such as predicting product sales or movie revenue from online reviews, we propose to construct explanatory path-based features based on the knowledge graph. The explanatory path-based features especially benefit from task-aware knowledge graph, because task-aware knowledge graph not only enlarges the coverage but also organizes the newly discovered knowledge in a disciplinary way.

To show the effectiveness of the proposed methods, we study one case of entity-centric classification. We propose to define the task awareness measurement function based on the relatedness of entities/relations/ attributes to the central entities, as well as some statistical analysis based on the co-occurrence between features and the central entities. The trimming of the knowledge graph can then be done according to the estimated task awareness values. The knowledge graph is then expanded by discovering new entities and relations based on the jointly embedding model and finding unknown aliases from the training examples. Our evaluation results show that the best way to construct the knowledge graph-based features for the entity-centric classification problem is to represent documents by the entities mentioned in it along with the context information. The trimming and the expansion of the knowledge graph both help to improve the performance, but the expansion is more helpful since the coverage problem impact the overall performance significantly.

Second, we study another case, of entity-centric regression. We use the same way to construct the task-aware knowledge graph as for the entity-centric classification task because to identify which text snippets is relevant to the central entity is usually the primary step to identify informative text information. Our study shows that when mapping the entity into its path to the central entities in the knowledge graph which explain the relationship between them, the features work better than using individual entities as features. In those path-based features, background knowledge is embedded in the feature space, which makes the features customizable to different types of entities. Meanwhile, the features are not limited to the existing entities and can also be generalized to unseen data, allowing it to better handle the emerging entities. Though the generated features are useful in general, they can be very noisy in some regression task especially when the correlation between the text and the response variables are really weak. To solve the problem, a two-step filtering approach is applied to further reduce the noise. In the first step, a t-statistic based measure is used to select potentially better-performing features. Features that work well on all the corresponding entities are likely to have high t-stats value and would be favored. In the later stage, we use the mixed-effect model to analyze where the impact of a feature comes from, which helps to further reduce noisy features. We evaluate the proposed method on two realistic scenarios of predicting revenues of unseen movies based on movie reviews. The experiment results show that our method is effective for finding good features, which helps to improve the prediction

accuracy significantly. The proposed method can also help reveal interesting interpretable features that can help explain changes of values of the target variable.

To summarize, the main contributions of this thesis include: (1) We systematically studied the knowledge graph-assistant feature engineering in text-based prediction. We presented several ways to construct the knowledge graph-based features and investigated their performance in different tasks. (2) We introduced the concept of task-aware knowledge graph and proposed a general framework for constructing and applying the task-aware knowledge graph to text-based prediction. We showed an instantiation of how to prune the knowledge graph to reduce noise and how to expand the knowledge graph to enlarge coverage in some specific types of applications. (3) We introduced a family of text-based prediction tasks called entity-centric prediction tasks and systematically studied effectiveness of many methods of constructing features for such tasks by leveraging knowledge graph. (4) We proposed and studied two general ways to expand knowledge graph. Experimental results show that both approaches are effective. Generally, when the coverage of the central entities in the existing generic knowledge graph is good enough, finding new entities and relations by the jointly embedding model is more promising. While when the coverage of the central entities is poor, making use of more general relations such as paradigmatic and syntagmatic relations is more beneficial. (5) We proposed task-specific instantiations of the general task-aware knowledge graph framework for two representative entity-centric tasks and show that task-aware knowledge graph not only substantially outperform task-independent knowledge graph but also even comparable to the manually-created features which are elaborately designed for the task.

This thesis is organized as follows:

Chapter 2 briefly reviews the literature.

Chapter 3 systematically studies knowledge graph-assistant feature engineering in entity-centric prediction tasks.

Chapter 4 introduces the method to expand knowledge graph by paradigmatic and syntagmatic relations, and examine whether the expansion can produce additional benefit.

Chapter 5 proposes a general framework to construct task-aware knowledge graph.

Chapter 6 instantiates the general framework proposed in Chapter 5 in entity-centric classification.

Chapter 7 presents how to exploit the task-aware knowledge graph constructed in Chapter 6 in another type application of entity-centric prediction – entity-centric regression. A two-stage filtering method is proposed to select the knowledge graph-based features.

Chapter 8 concludes this thesis and discusses the future work.

CHAPTER 2: LITERATURE REVIEW

Construction of knowledge graph has received great attention from both academia and industry ever since the knowledge graph theory was introduced. Early work usually involved much human effort like ontologies in different languages (e.g., the early version of WordNet [6]). Expert-edited knowledge graph is of high quality, but the construction and maintenance are very expensive and labor-intensive, making it not scalable.

The automatic construction of knowledge graph helps solve those problems and thus has become increasingly popular. Due to the rapid development of related techniques such as text mining and natural language processing, recent years have witnessed a proliferation of large-scale or even web-scale knowledge graphs being generated in an automatic way. Semi-structured text data such as Wikipedia infobox is widely used in the construction of knowledge graph. For example, DBpedia [7], Freebase [8] and YAGO [9] are all built from such kind of semi-structured text data.

Though semi-structured text data is better organized which is easier to exploit, a much richer resource still lies in the unstructured free-form text data. If we can make use of the free-form text data, we can leverage the entire web to construct a powerful knowledge graph. A lot of existing work has been explored in this direction. For example, NELL [10], PROSPERA [11], DeepDive [12] and Knowledge Vault [13] are all generated from the entire web and they do not require the data source to be structured.

Besides the generic knowledge graph, extensive effort has also been devoted to the construction of domain knowledge. For example, social domain knowledge graphs that can discover emerging entities [14]; health knowledge graph built from electronic medical records [15] and web source [16]; enterprise knowledge graph that provides useful information about companies [17]; knowledge graph for technical and scientific domain [18], [19]; and knowledge graph for mobile apps [20]. Compared to the generic knowledge graphs, the domain knowledge graph can provide tailored background knowledge in a particular domain. However, they are domain-oriented rather than task-oriented, which can not guarantee the coverage of the background knowledge in a random task even in the specific domain. For example, a knowledge graph constructed for the movie domain is still not likely to includes all the aliases of all the movies and other relevant entities in an arbitrary document, especially when the language is quite informal (e.g. reviews from tweets and other web forums).

There is also exploration of new schema of knowledge graph, such as event-centric knowledge graph [21, 22], visual knowledge graph that contains both visual and textual entities [23], temporal knowledge graph which learns dynamic representation of entities over time [24].

These new schemas are not limited to text representations. Rather than representing the relations between entities in a static network, they also explored the possibility of taking multi-media information as supplement or even as the key points, as well as the possibility of using dynamic network to timely update the network structure in a much more flexible way.

Both the expert-edited and the automatically-generated knowledge graphs are important resources to help machines acquire human knowledge; they are widely used in many real-world applications such as question answering [25, 26], opinion mining [27, 28] and query expansion [29, 30].

Though great achievement has been made, the construction and application of knowledge graph in real applications are far from optimal. Most existing knowledge graphs are generated for all purposes, but not designed to provide customized knowledge for a specific task. Their construction is thus not task-dependent and generally cannot fully satisfy the knowledge need in a particular task. Compared to the construction of existing knowledge graphs, the way we generate a task-aware knowledge graph is not only task-oriented but also enables the knowledge graph to learn new knowledge from the application. Conceptually, what we propose is to tightly integrate construction of a relevant knowledge graph to a task and its application in the task, thus enabling exploitation of feedback from task performance to iteratively optimize the knowledge graph for the particular task.

Note that the construction of task-aware knowledge graph is not incompatible with the traditional way of generating a general knowledge graph. Both schema-less and pattern-based methods that are used for the construction of general knowledge graph can also be utilized in the expansion step of constructing a task-aware knowledge graph. Knowledge graph completion and relation extraction methods [31, 32, 33] can also be used to enhance the task-aware knowledge graph when the coverage of the existing knowledge graph is not enough. Besides, existing general knowledge graphs can be utilized to initialize the task-aware knowledge graph to provide basic background knowledge.

Knowledge graph is leveraged to benefit a variety of applications, among which text-based prediction is a representative one. Text-based prediction is a powerful tool to predict real-world phenomenon from text data. It has been explored in many practical applications such as election result prediction ([1, 2]), health research ([34, 35]) and product sales prediction ([3, 5, 4]). However, the existing methods have not fully exploited knowledge graph to construct effective and interpretable features for prediction.

Free-form text is sometimes not informative enough for complex prediction tasks, and semi-structured text is more popularly used as a result. For example, hashtag and user information are useful features when leveraging tweets [2]. Another widely-used intermediate

feature from free text is sentiment. For example, a study has shown that the movie revenue performance is somehow correlated with sentiment polarity of the reviews [36]. Such kinds of meta features or intermediate features inferred from free text can be useful for a certain type of task. However, they are not universally applicable to different domains. For example, sentiment could be a powerful feature for sales prediction but it may not work so well for health decision-making system. Besides, manually designed features need to be carefully selected in advance and it is hard to exhaust all kinds of useful features by hand. Compared to most of the existing methods, our method discovers correlation between features and response in an automatic way and can leverage the increasingly available knowledge graph resources.

As an important method of knowledge representation, knowledge graph has been widely used in various text mining applications, such as document similarity computation [37, 38], topic discovery [39, 40], opinion mining [27, 28] and information retrieval [41, 42]. Background knowledge provided by KGs can be leveraged in various ways. Taking document representation as an example, it can be enriched by knowledge graphs at either entity level (e.g. [43]) or sub-graph level (e.g. [37, 41]). In this thesis, we make use of both entity-level and relation-level knowledge and incorporate it with contextual information. This is especially helpful when the correlation between the textual information and the response variable is weak, which requires a tight collaboration of local context with background knowledge.

The proposed method mainly focuses on finding better features for entity-centric text regression applications. However, The proposed two-stage filtering algorithm is generalizable to any text-based prediction tasks. An important complementary way to enhance the performance is to improve the regression model itself. When the response is continuous variable, linear regression is typically employed. One problem with linear regression model is that the correlation between the output response and the input features is much more complex and non-linear model might be more proper [44]. The improvement of learning model can always be in parallel with generation of better features, and the combination of them would be beneficial in general for all kinds of applications. Our method can thus be potentially combined with many different regression models to support a variety of real-world applications.

CHAPTER 3: KNOWLEDGE GRAPH-BASED FEATURE ENGINEERING IN ENTITY-CENTRIC PREDICTION

3.1 INTRODUCTION

Most explicit knowledge is stored in the form of unstructured textual information, which makes the organizing, analyzing and exploiting of the ever growing documents on the Internet a challenging and vitally important problem. These text mining and management applications usually include the procedure of preprocessing, featurization and finally being casted as an unsupervised or supervised learning problem which produces some response variables according to the text input. In this procedure, feature engineering is critical to the performance of many applications. Lexical, syntactic and semantic features have thus been studied in previous studies.

Lexical features like bag of words have achieved great success in many text mining and management tasks and become the most popular way for text representation. In a variety of text-based tasks, lexical features are powerful in the detection of correlation between the text content and the response variable. For instance, sentimental words indicate the sentiment polarity in the task of analyzing the sentiment of natural language text, and topic words differentiate thematic categorization of news article about business, technology, entertainment, etc.

Though lexical features have been shown to be effective and robust, they have their limitation of modeling the textual information accurately. On one hand, they ignore the syntactic information. The structure of natural language, though not necessarily helpful in some cases, is needed in many tasks such as non-native text analysis where the syntactic structure helps to understand how the speakers construct their sentences [45].

On the other hand, lexical features are not able to capture the semantic relation between words. For example, they are not able to distinguish synonyms, which makes the corresponding dimensions non-orthogonal in the feature space. To solve this problem, a common solution is to represent the documents by low-dimensional vectors in a latent semantic space such as topic space and word2vec embedding space. The latent semantic space captures the semantic relations among words, thus models the textual information more accurately, especially when diversified vocabulary is used to express similar content.

Though syntactic and semantic features can break through the limitation to some extent, they are still not capable of dealing with more challenging tasks where response variables cannot be directly derivable from the text. Syntactic information is helpful for understanding the language structure, but is lack of deeper semantic information. The improvement of

text representation brought in semantic features is mainly limited to the scope of internal relations within text contents, rather than directly reveal how the text content is related to the response variables. For example, if we want to identify the online reviews about certain movies, it is more difficult to get the signal features compared to finding discriminative words or phrases in thematic categorization tasks. In such kind of tasks, the response variable is distantly related to what is explicitly mentioned in the text, which means that there exists a semantic gap. The textual information is related to the response variables by more complex reasoning with background knowledge, thus surface features extracted from the text are usually not effective in bridging the gap.

Among all the applications where background knowledge is required to allow people to see the pattern behind the textual information, a large number of them are entity-based where the response variables are closely related to certain entities. For example, in many text classification or regression tasks, the response variables can be a group of entities or a certain attribution of them. We refer to such kind of application as entity-centric text application, and name the entities that the response variables are related to as central entities. For instance, identifying online reviews about movies, predicting the sales performance of products, are both entity-centric text applications. In the movie review classification task, the central entities are the movies and the response variables can be their IDs. While in the sales prediction task, the sales performance can be considered as an attribute of a product and the product are the central entities.

Though semantic gap also exists in entity-entric text applications, the natural connection between the response variables and the central entities makes it easier to capture the correlation between textual information and the response variable with assistance of knowledge graph. In order to bridge the gap, we propose to leverage knowledge graph as an auxiliary to represent documents. Background knowledge provided by knowledge graph helps to distinguish informative textual signal from the noise, thereby better exploiting the predictive power of the textual information.

3.2 DEFINITION

Before introducing our exploration of exploiting knowledge graph for entity-centric prediction in details, we first formulate the problem and define the concepts that will be used in the thesis.

3.2.1 Knowledge graph

A generic knowledge graph is composed of entities and relations between them. An entity or a relation in a knowledge graph can have an assortment of attributes, e.g., name and aliases, types, etc. Formally, we have the following definition for entity and relation:

Definition 3.1. ***Entity:*** An entity is an object or concept that can be distinctively identified. It has a set of attributes which can be considered as its components or properties. Formally, let e be an entity then its attribute set is denoted as $e.A = \{e.A_1, e.A_2, ...\}$ where $e.A_i$ is the i-th attributes of e.

Definition 3.2. ***Relation:*** A relation is the relationship between two entities, and it can be represented as a triple $r = \langle e_i, r.A, e_j \rangle$. Here e_i and e_j are the two entities that are connected by the relation r. We name e_i and e_j as the head entity and the tail entity connected by relation r respectively. $r.A$ is the attribute set of r.

Example 3.1 shows an instance of entity. It is a movie entity contains attributes such as name and budget. We define a function $\mathcal{K}(a)$ to obtain the key of an attribute a. $\mathcal{V}(a)$ is the function to obtain its value. $\forall a = \langle k, v \rangle, \mathcal{K}(a) = k, \mathcal{V}(a) = v$. In this example, we have $\mathcal{K}(\langle name, LaLaLand \rangle) = name$ and $\mathcal{V}(\langle name, LaLaLand \rangle) = LaLaLand$.

Example 3.1. A movie entity in Figure 1.2 can be represented as $e_1 = \{\langle name, LaLaLand \rangle, \langle Budget, \$30million \rangle, ...\}$. It contains attributes like name, budget, etc. Each attribute is represented as a key-value pair where the key is the type of the attribute (e.g., name, budget).

An example of relation is given in Example 3.2. Type information and confidence scores are popular attributes for relation. In this example, the relations contain only one attribute which is the type of the relation.

Example 3.2. As shown in Figure 1.2, the movie entity e_1 are connected with two person entities: (1) its director by relations such as $\langle e_1, \langle type, DirectedBy\rangle, e_2\rangle$; (2) and the actress who starred in the movie $\langle e_1, \langle type, StarredBy\rangle, e_3\rangle$.

Based on the definition of entity and relation, we can formally define a generic knowledge graph as:

Definition 3.3. ***Knowledge Graph (KG):*** A knowledge graph G is composed of entities and relations between them. Formally, $G = \{E, R\}$ where E and R are the entity set and relation set respectively.

Entities in the knowledge graph can be considered as nodes, while relations between them can be treated as edges. Based on the graph structure, we can define the path between two entities:

Definition 3.4. ***Path between Entities:*** The set of all possible paths between two entities in knowledge graph G is $path(e_i, e_j, G) = \{\langle r_1, r_2, \dots r_l\rangle | r_1 = \langle e_i, r_1.A, e_{k1}\rangle, r_2 = \langle e_{k1}, r_2.A, e_{k2}\rangle, \dots, r_l = \langle e_{k,l-1}, r_l.A, e_j\rangle\}$.

Example 3.3. A path between e_1 and e_4 in Figure 1.2 is: $\langle\langle e_1, \langle type, DirectedBy\rangle, e_2\rangle, \langle e_2, \langle type, Spouse\rangle, e_4\rangle\rangle$

The distance between two entities in a knowledge graph is then defined as the length of the shortest path between them.

Definition 3.5. ***Distance between Entities:*** The distance between two entities in knowledge graph G is defined as $dis(e_i, e_j, G) = \min_{p \in path(e_i, e_j, G)}\{len(p)\}$, where $len(p)$ is the length of path p.

For instance, the distance between e_1 and e_4 is 2, as the path shown in Example 3.3 is the shortest path between them.

3.2.2 Task Description

Text-based prediction takes free-form text data as input and produces an estimate of the response variable as output. Formally, we represent the input text as a set of documents $D = \{d_1, d_2, \ldots, d_n\}$ and the response variable values as a vector Y. The text-based prediction task is thus essentially a mapping from D to Y: $\mathcal{F}(D \to Y)$.

Definition 3.6. ***Document:*** A document d is a set of sentences $d = \{s_1, s_2, \ldots, s_{|d|}\}$ where each sentence can be represented as a sequency of words $s_i = \langle w_{i_1}, w_{i_2}, \ldots, w_{i_{|s|}} \rangle$.

Definition 3.7. ***Response Variable:*** The response variables are the output of text-based prediction, and can be represented as a vector $Y = [y_1, y_2, \ldots, y_n]^T$ where y_i is the response variable of the i-th document d_i in D. All the y's are numerical, either discrete or continuous.

Entity-centric prediction is a family of text-based prediction tasks where the response variables are the attributes of a group of entities. This kind of prediction tasks is very common, including examples such as prediction of sales of products, revenues of movies, stock prices, and poll results of presidential candidates.

Definition 3.8. ***Entity-centric Prediction:*** A text-based prediction task t takes Y as its response variables. If each response variable $y \in Y$ is associated with an attribute of an entity $(\forall y \in Y, |\{e|e \in E \wedge y \in e.A\}| = 1)$, the task t is an entity-centric prediction task.

Definition 3.9. ***Central Entity:*** Given an entity-centric prediction task t whose response variables are Y, entities that are associated with its response variables are named as central entities. We denote the central entities of t as $t.E_c$, then $t.E_c = \{e|e \in E \wedge (\exists y \in Y \;\; s.t. \;\; y \in e.A)\}$.

Example 3.4. If we want to collect customers' review about some products from social media and classify them based on the product being reviewed, it can be considered as an entity-centric prediction task. The products are the central entities. The response variables are the class labels which equals to the IDs of the products.

Example 3.5. A movie entity has various attributes such as name, release date and box office (e.g., $e = \{\langle$*Name: La La Land*$\rangle$, $\langle$*Release date:12-31-2016*$\rangle$, $\langle$*Box office: $30 million*$\rangle, \ldots\}$), and the response variable (i.e., box office) is one of them. If the task is to predict the box office for a given set of movies from online reviews, it is an entity-centric prediction task with the movies as central entities.

Among all kinds of entity-centric predictions, we mainly focus on two fundamental types – entity-entric classification and entity-centric regression. Example 3.4 is an example of entity-centric classification, which is a fundamental family of entity-centric prediction. Entity-centric classification is a representative task of entity-centric prediction, where the central entities are directly connected to the output. In other kinds of entity-centric tasks, there may be additional information required in the outcome besides central entities. For example, However, the primary step in many real-world entity-centric prediction tasks is usually to distinguish the relatedness between a central entity and an output variable. In this sense, entity-centric classification is helpful for many other kinds of entity-centric text analysis.

The response variables in entity-centric are discrete. While in some other types of tasks, the response variables can be continuous, such as the entity-centric regression task shown in Example 3.5. Such tasks can sometimes be much more challenging the entity-centric classification tasks, especially when the response variables are weakly correlated with the input text.

3.3 KNOWLEDGE GRAPH-ASSISTANT FEATURE EXTRACTION

In many text classification or regression tasks, a popular way to featurize the text data is to use lexical features such as bag of words. When using individual words or phrases as features, one issue is that it ignores the semantic relations between features. For example, it can neither distinguish polysemous words nor recognizing synonyms. This problem can be

alleviated by mapping documents to feature vectors in a latent semantic space. However, it still does not go beyond the surface semantic level from the perspective of capturing the correlation between text input and response variables.

In entity-centric classification or regression tasks, the central entities could play the role of the bridge linking the input text and the response variables which are their attributes. It is likely that the textual information has ties to the response variables if the content is related to some of the central entities. For example, if the task is to classify online reviews according to the mentioned movies, we can identify a sentence like "*I love the movie and City of Stars is my favorite song of the year*" as the review about the movie "La La Land" because we know that "City of Stars" is one of the songs in "La La Land".

In general, entities related to central entities mentioned in the text content indicate a possible association with the response variables. Meanwhile, the relationship between the mentioned entity and the central entity casts into the insight how the textual information is tied to the response variables. Besides, the contextual information may also of great value to gain additional information. Our goal is to leverage these desirable features in the document representation and deliver them to the classifier or regressor in an effective form. With the assistance of knowledge graph, we can obtain the related entities and their corresponding relations easily. Therefore, we propose to featurize the text in a knowledge graph-assistant way. In order to find a better way to construct features based on knowledge graph, we propose four approaches for feature extraction.

3.3.1 Entity-based featurization

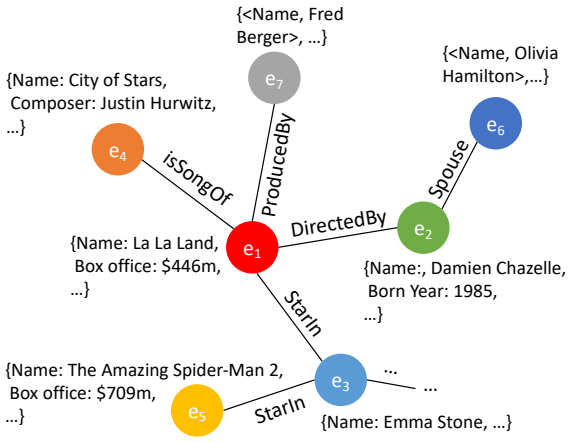
Given a knowledge graph, we first find entities that are closely related to the central entities. Entities that are connected by a certain type of relations or paths in the knowledge graph are related to each other in a particular way. Generally, they are more closely related when the path is shorter. Based on this assumption, we construct a sub-graph formed by the central entities and their close neighbor entities together with the relations between them. Intuitively, this subgraph represents the relevant knowledge from the KG to our prediction task.

Formally, let $G_0 = \{E_0, R_0\}$ be the original KG. The extracted subgraph is defined as $G = (E, R)$, with

$$E = t.E_c \cup \{e \in E_0 | \exists e' \in t.E_c \ s.t. \ dis(e, e', G_0) \leq \theta_d\} \tag{3.1}$$

$$\begin{aligned} R = \{r \in R_0 | \exists e \in t.E_c, e' \in E \ s.t. \ r \in p \wedge p \in path(e, e', G_0) \\ \wedge len(p) = dis(e, e', G_0)\} \cup \{\langle e, SELF, e\rangle | e \in E\} \end{aligned} \tag{3.2}$$

Figure 3.1: An example of related entities in the knowledge graph.



where $t.E_c$ are the central entities of the task t. $path(e, e', G_0)$ is the set of all possible paths from entity e to e' in the knowledge graph G_0 and $dis(e, e', G_0)$ is the distance between the two entities which equals to the length of the shortest path. $len(p)$ denotes the length of the path p. The neighbor entities can be directly connected to one of the central entities by a relation, or they can be connected by a path whose length is no larger than θ_d. Note that we add an extra relationship "*SELF*" connecting an entity with itself, which will be used later.

Example 3.6. As the example shown in Figure 3.1, the task is to classify online reviews according to the mentioned movies. There are a bunch of movie entities to be identified, including e_1 ("La La Land") and e_5 ("The Amazing Spider-Man 2"). The entity set of the subgraph is $\{e_1, e_2, e_3, e_4, e_5\}$ when we set the cut-off threshold of path length (θ_d) as 1. If we set θ_d to be 2, then e_6 will be included in the entity set of the subgraph as well.

Given a sentence s, how should we derive features based on the subgraph G? Intuitively, we first want to check whether any of the entities in G occurs in s, and if so, those matched entities in s would be relevant to the prediction task and thus should be the basis for constructing features. A most straightforward way to featurize text based on these entities is to directly use the entities themselves as the feature space. A document will be then represented as the matched entities in it, as the example shown in Example 3.7. We name the features generated by this featurization method as *entity-based features*.

Formally, the entity-based features extracted from a sentence s are defined as:

$$\mathcal{E}(s) = \{e | e \in E \wedge v \in name(e) \wedge v = \langle w_1, \ldots, w_k \rangle \wedge s = \langle w'_1, \ldots, w'_{|s|} \rangle \\ \wedge w'_j = w_1 \wedge \ldots \wedge w'_{j+k-1} = w_k \wedge j \geq 1 \wedge j + k - 1 \leq |s|\} \tag{3.3}$$

where $name(e)$ is the function to return the names of the entity e including all its aliases: $name(e) = \{\mathcal{V}(a) | a \in e \wedge \mathcal{K}(a) = name\}$.

Example 3.7. Taking the document "*I loved La La Land so much. City of Stars is my favorite song of the year*" as an example, it is represented by the following vector given the subgraph shown in Figure 3.1 when using entity-based featurization:

$$\begin{array}{c} \quad e_1 \quad e_2 \quad e_3 \quad e_4 \quad e_5 \quad \ldots \\ d \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & \ldots \end{pmatrix} \end{array}$$

We find two entities mentioned in the documents: e_1 ("La La Land") and e_4 ("City of Star").

3.3.2 Context-aware entity-based featurization

The related entities help to discover useful information in the text content. However, if we only use such entities as features, we would lose much context information about the mention of an entity. Apart from the occurrence of related entities, the contextual information is also helpful. In the example shown in Section 3.7, if we only want to know which movie the review "I love La La Land and City of Stars is my favorite song of the year" is talking about, the occurrence of "La La Land" and "City of Stars" provides enough information to infer that it is about the movie La La Land. However, if we also want to analyze the sentiment, it is hard to predict the sentiment polarity if the only information we can get is the occurrence of entities. In this case, what leads to a deeper understanding of the sentiment of the sentence is content like "I loved La La Land so much".

In fact, context is helpful in many cases, it together with the mentioned entity tells us more detailed information. To capture the context information, we thus define our *context-aware entity-based feature* as a 3-tuple representing a combination of a mentioned relevant

entity with any n-gram from the "left context" and any n-gram from the "right context" of the mentioned entity.

To make such features more comparable with each other, we control the total length of the two "contextual" n-grams with a parameter l (i.e., requiring their total length to be l). Besides, to ensure relevance of the contextual n-grams, we further restrict the size of the context window on both sides of the entity to be not exceeding another parameter W.

Formally, given an entity e, the context-aware entity-based feature set for sentence s with parameters l and W is defined as:

$$\begin{aligned}\mathcal{C}(s,e,l,W) = \{(\langle w'_i,\ldots,w'_{i+l_1}\rangle, e, \langle w'_q,\ldots,w'_{q+l_2}\rangle) | v \in name(e) \wedge v = \langle w_1,\ldots,w_k\rangle \\ \wedge\, s = \langle w'_1,\ldots,w'_{|s|}\rangle \wedge w'_j = w_1 \wedge \ldots \wedge w'_{j+k-1} = w_k \wedge i + l_1 < j \wedge j - i \leq W \\ \wedge\, q + l_2 \leq |s| \wedge j + k - 1 < q \wedge q + l_2 - (j + k - 1) \leq W \wedge l_1 + l_2 = l\}\end{aligned} \tag{3.4}$$

We can use multiple contextual phrase lengths (l) to generate the features. Denote the set of all the contextual phrase lengths we want to have as L. A document will then be represented by all the context-aware entity-based features that are extracted from it:

$$\mathcal{C}(d,l,W) = \cup_{s\in d, e\in E, l\in L}\mathcal{C}(s,e,l,W) \tag{3.5}$$

where E is the entity set of the knowledge graph that is used to extract the entity-aware entity-based features.

Example 3.8. Given a document "*I loved La La Land so much. City of Stars is my favorite song of the year*" and the knowledge graph shown in Figure 3.1. The document is represented as the following vector when we set both l and W as 2:

$$\begin{array}{c} \quad (\langle \text{I love}\rangle, e_1, \langle\rangle) \quad (\langle \text{love}\rangle, e_1, \langle \text{so}\rangle) \quad (\langle\rangle, e_4, \langle \text{is my}\rangle) \quad \ldots \\ d\,\Big(\quad 1 \qquad\qquad 1 \qquad\qquad 1 \qquad\quad \ldots\Big) \end{array}$$

3.3.3 Explanatory path-based featurization

Intuitively, entity-level features are semantically relevant to the prediction task, but they tend to cause data sparsity problem and cannot adapt to unseen data such as newly-emerging

entities. To address these limitations, we propose a method to transform the entity-level features to a *explanatory path-based features* that involve both entities and relations.

More specifically, we transform the entity in the entity-level features to its relationship with the central entities. This transformation is a novel idea in feature construction, which generalizes the original entity-level features without sacrificing relevance. Indeed, the relation provides an "explanation" why the entity can be potentially useful for the prediction task. Thus it generalizes much better than the original entities and is also more interpretable. Since entities that are related to the central entities in the same way can be expected to play a similar role in the prediction, merging them together as one (abstract) feature not only alleviates the sparsity problem in the feature space, but also ensures the needed generalization when we encounter entities not seen in the training data, because relation types are much more prevalent than individual entities.

Given an entity e and a set of central entities $t.E_c$, the explanatory path are defined as:

$$\Omega(e, t.E_c) = \{(\mathcal{T}(p), e') | e' \in t.E_c \wedge p \in path(e, e', G) \wedge path(e, e', G) = dis(e, e', G)\} \quad (3.6)$$

where $\mathcal{T}(p)$ is a function to get all the relation types in path p:

$$\mathcal{T}(p) = \langle \mathcal{V}(r_1.A_1), \mathcal{V}(r_2.A_1), ...\mathcal{V}(r_l.A_1) \rangle \text{ if } p = \langle r_1, r_2, \dots r_l \rangle \quad (3.7)$$

Here we assume that each relation r has type information as one of its attributes, and denote the corresponding attribute as $r.A_1$ (the first element in the attribute set). Note that if the entity e is one of the central entities, then $\mathcal{T}(p) = \langle \text{"}SELF\text{"} \rangle$.

Given a sentence s and an entity e, the explanatory path-based features are defined as:

$$\mathcal{P}(s, e) = \begin{cases} \Omega(e, t.E_c) & \text{if } e \in \mathcal{E}(s); \\ \varnothing & \text{otherwise.} \end{cases} \quad (3.8)$$

Finally, a sentence s is represented by all the explanatory paths that can be found in it:

$$\mathcal{P}(s) = \{\mathcal{P}(s, e) | e \in E\} \quad (3.9)$$

Example 3.9. Take the document *"Emma Stone's performance hits high notes. City of Stars is my favorite song of the year"* as an example. Again, we have the knowledge graph shown in 3.1. Thera are two

central entities – "La La Land" (e_1) and "The Amazing Spider-Man 2" (e_5). We first extract the related entities mentioned in the document, namely "Emma Stone" (e_3) and "City of Stars" (e_4). In the next step, we find their explanatory paths to the central entities, i.e., $\{\langle(e_3, \text{starIn}, e_1)\rangle, \langle(e_3, \text{starIn}, e_5)\rangle\}$, $\{\langle(e_4, \text{isSongOf}, e_1)\rangle\}$ for e_1 and e_4 respectively. Finally, we ignore the mentioned entity itself and utilize the rest link struction as features, which are $\{\langle(\text{starIn}, e_1)\rangle, \langle(\text{starIn}, e_5)\rangle\}$ and $\{\langle(\text{isSongOf}, e_1)\rangle\}$. These features are used to represent the document:

$$d \begin{pmatrix} \{\langle(\text{starIn}, e_1)\rangle, \langle(\text{starIn}, e_5)\rangle\} & \{\langle(\text{isSongOf}, e_1)\rangle\} & \dots \\ 1 & 1 & \dots \end{pmatrix}$$

By mapping the related entities to their explanatory paths, entities that are related to the central entities in a similar way can be grouped together. Note that if we keep the related entities in the explanatory paths, the formed feature space is equal to or larger than the entity-based feature space introduced in 3.3.1, which may even exacerbate the sparsity problem.

3.3.4 Context-aware explanatory path-based featurization

Though explanatory path-based featurization helps to reduce the degree of freedom by merging entities playing a similar role together in the feature space, it may also suffer from the information loss problem. For example, we still cannot identify the semantic polarity towards the song when we see the sentence "City of Stars is my favorite song of the year." by using the explanatory path-based features even if we are aware that City of Stars is a song of La La Land. This is mainly due to the fact that the context information is missing. A solution to this problem is again to combine the context information.

Similar to the context-aware entity-based featurization, we also include the context words and construct so-called *context-aware explanatory path-based features*.

Formally, given an entity e, the context-aware explanatory path-based feature set for

sentence s with parameters l and W is defined as:

$$\begin{aligned}\mathcal{M}(s,e,l,W) = \{(\langle w'_i,\ldots,w'_{i+l_1}\rangle, \Omega(e,t.E_c), \langle w'_q,\ldots,w'_{q+l_2}\rangle) | v \in name(e) \wedge v = \langle w_1,\ldots,w_k\rangle \\ \wedge\, s = \langle w'_1,\ldots,w'_{|s|}\rangle \wedge w'_j = w_1 \wedge \ldots \wedge w'_{j+k-1} = w_k \wedge i + l_1 < j \wedge j - i \leq W \\ \wedge\, q + l_2 \leq |s| \wedge j + k - 1 < q \wedge q + l_2 - (j + k - 1) \leq W \wedge l_1 + l_2 = l\}\end{aligned} \tag{3.10}$$

We can have multiple contextual phrase lengths (L) and the final context-aware entity-based features extracted from the document are:

$$\mathcal{M}(d,l,W) = \cup_{s\epsilon d, e\epsilon E, l\epsilon L}\mathcal{M}(s,e,l,W) \tag{3.11}$$

Example 3.10. Given the knowledge graph shown in 3.1, the sentence *"City of Stars is my favorite song of the year"* will be represented by the following vector based on context-aware entity-based featurization by setting both l and W to be 2:

$$d\begin{pmatrix} (\langle\rangle, \{\langle(\text{isSongOf}, e_1)\rangle\}, \langle\text{is my}\rangle) & \ldots \\ 1 & \ldots \end{pmatrix}$$

3.4 EXPERIMENT

In this section, we apply our method in two applications: (1) classifying reviews according to the mentioned movies; (2) predicting the revenue of movies based on reviews.

3.4.1 Dataset and experiment setup

We use the dataset in [5] to conduct our experiment. The dataset contains reviews of 1,718 movies released from 2005 to 2009. The reviews are collected by crawling Meta-Critic[1]. In the review classification task, we remove reviews with no more than 10 words because it is often hard to infer the referred movie from the short reviews. For example, reviews like "I love the movie" do not provide enough information for us to identify the movie labels.

[1] www.metacritic.com

Movies with no less than 12 reviews are remained in the dataset for the classification task. Finally, a dataset contains 1,354 movies are used, and these movies are considered as the class labels. 10-fold cross validation is used in this study to evaluate the performance of various features in entity-centric classification.

For the revenue prediction task, all the reviews from the original dataset are used. They are partitioned temporally into training, development and test set. Movies released from 2005 to 2007 are used for training; and those released in 2008 are included in the development set. Based on the history data, the future (2009) revenue performance is to be predicted. The central entities in the test set are all newly-emerging ones, reflecting the prediction task in a real-world application very well.

A logistic regression model with L-1 regularization is trained to identify the class labels for the movie review classification task. While a linear regression model with L-1 and L-2 regularization is trained to predict the movie revenue.

Four metrics are used to evaluate the performance of features in the classification tasks, namely, accuracy, precision, recall and F1 measure. To assess the performance of features in the revenue prediction task, we adopt two metrics – mean absolute error (MAE) and symmetric mean absolute percentage error (SMAPE). The symmetric mean absolute percentage error is calculated as:

$$SMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|F_i - A_i|}{|F_i| + |A_i|}$$

where A_i is the actual revenue of the i-th movie and F_i is the predicted revenue. n is the number of movies in the test set.

We employ two different knowledge graphs in our experiment. One is YAGO [9]. The other is constructed from the infobox of the corresponding Wiki page about the movies, merging the meta information such as authors, actors/actresses and directors of each movie provided by the dataset in [5].

3.4.2 Feature engineering in the entity-centric classification task

The primary goal of our study is to see what kind of features works best for the entity-centric classification and regression, and whether knowledge graph-assistant featurization is useful for such kind of applications.

We investigate four families of featurization methods. The first one is lexical features, including different kinds of n-gram features. The second one is syntactic features such as POS tags and dependency triples. The third one is semantic features. The named entity identified from the text are used to form the feature space. We also apply word embedding

Table 3.1: Performance in classification task

feature	accuracy	precision	recall	F-1 measure
unigram	0.374	0.371	0.362	0.366
uni- and bigram	0.378	0.379	0.365	0.372
uni-, bi- and trigram	0.380	0.376	0.367	0.371
POS	0.270	0.249	0.243	0.246
DEP	0.171	0.158	0.152	0.155
NER	0.327	0.448	0.323	0.376
W2V	0.193	0.181	0.177	0.179
Wiki (EF)	0.339	0.419	0.332	0.370
Yago (EF)	0.303	0.439	0.296	0.353
Wiki (CaEF)	**0.532***	**0.511***	**0.513***	**0.512***
Yago (CaEF)	0.451	0.437	0.432	0.435
Wiki (ExPF)	0.230	0.364	0.226	0.278
Yago (ExPF)	0.217	0.348	0.212	0.263
Wiki (CaExPF)	0.458	0.451	0.441	0.446
Yago (CaExPF)	0.307	0.311	0.291	0.301

Two-tailed t-test is done for paired data. In each pair, one is the method that get best performance, and the other one is any of the other methods. * indicates p-value < 0.01 for all tests.

technique to represent documents in a latent semantic space. The last one is the proposed knowledge graph-based features.

The performance of different families in the entity-centric of features are summarized in Table 3.1. The first three rows are n-gram features. "POS" means n-grams (including unigram, bigram and trigram) with their POS tags. "DEP" is the dependency triples. "NER" is the named entity extracted from the text. The POS tagging, dependency triples and named entities are obtained by using Stanford NLP tools. Word vectors pre-trained on Common Crawl [46][2]. are used to represent documents in the latent embedding space. The document vector is calculated by taking the average of the word vectors. The four methods for knowledge graph-based feature construction proposed in Section 3.3.1 are applied in both of the two knowledge graphs. "EF" indicates the entity-based featurization proposed in Section 3.3.1. "CaEF" represents the context-aware entity-based features (see Section 3.3.2). "ExPF" extracts the explanatory path-based features from text, as shown in Section 3.3.3. "CaExPF" is the context-aware explanatory path-based feature extraction method introduced in Section 3.3.4.

[2]The pre-trained vectors are downloaded from https://fasttext.cc/docs/en/crawl-vectors.html

We can see that it is hard to find the correct label when we only use n-gram features. This is because there are too many noisy features in n-grams. The n-gram based method is good at finding topic-level features. For example, "magic" is among the top features for all the three Harry Potter movies in the data set. It can help to find movies in the same genre. But it is not a powerful feature for finer-grained categorization such as to identify the referred movie. Such kind of features finally cause overfitting problems. For example, the accuracy can reach 1.0 on a subset of the training data but is much worse on the test data. Feature selection methods can help to alleviate the problem a little, but not fully solve it.

In general, lexical features work better than the syntactic features. The syntactic information does not help much in this entity-centric classification problem. The syntactic features are much sparser than the lexical features, which is a possible reason why the prediction accuracy decreases. Entity names are informative signals of the class labels, thus "NER" works the best among all the other features except knowledge-graph based ones. Our knowledge graph-based features significantly outperform others. Entities are more discriminative than relation information in the classification task, thus mapping entities to a subgraph consists of the explanatory paths does not help much.

Context information provides valuable signals to identify the class labels. When the context information is combined with the knowledge graph-based features, it can even work better, for both the entity-based ones and the explanatory path-based ones.

3.4.3 Feature engineering in the entity-centric regression task

In Section 3.4.2, we show that leveraging knowledge graph can provide tangible benefits to entity-centric classification tasks. A further question is that whether the knowledge graph-based features are helpful in more general entity-centric regression tasks. Entity-centric regression task can sometimes be much more challenging than classification task when the response variable that is to be predicted is very weakly correlated to the text information. We conduct the experiment to predict revenue performance based on movie reviews. The experiment includes two kinds of response variables, one is the weekend revenue and the other is per-screen revenue. The results are shown in Table 3.2 and Table 3.3. There are two models used to make the prediction. "LR" represents the linear regression model and "Elastic net" is the linear regression model with L-1 and L-2 regularization. Similar to the features used in the classification task, we also employ lexical (n-grams), syntactic (POS and DEP), and semantic (W2V) features in the regression task. "W2V" is calculated by a linear combination of the pre-trained word vectors using TFIDF weighting.

The results again show that the prediction task suffers a lot from the overfitting problem.

Table 3.2: Performance in prediction task with linear regression model

feature	Weekend revenue		Per-screen revenue	
	MAE($M)	SMAPE (%)	MAE($K)	SMAPE (%)
unigram	9.41	70.71	7.07	38.13
uni- and bigram	**9.03**	68.78	6.70	33.61
uni-, bi- and trigram	9.05	**68.55**	**6.68**	**33.50**
POS	9.11	70.04	7.12	37.66
DEP	9.40	69.27	6.91	35.17
W2V	10.04	**68.55**	8.86	46.83
Wiki(EF)	15.99	82.45	8.71	41.97
Yago(EF)	18.61	85.20	11.45	54.65
Wiki(CaEF)	10.78	69.90	7.33	36.49
Yago(CaEF)	10.44	72.96	7.13	36.68
Wiki(ExPF)	19.27	72.64	18.04	57.45
Yago(ExPF)	26.41	73.05	35.05	72.11
Wiki(CaExPF)	10.83	69.93	7.18	36.49
Yago(CaExPF)	10.43	73.06	6.90	35.29

Two-tailed t-test is done for paired data. In each pair, one is the method that get best performance, and the other one is any of the other methods. * indicates p-value< 0.01 and † indicates p-value < 0.05.

This is somewhat expected due to the large semantic gap between such surface features and the target variable. Taking unigram features as an example, we test the linear regression model trained for weekend revenue on a subset of the *training* set, and MAE is $16.54, much less than what we get for the test set (over $9 million). When we utilize NLP tools to get more sophisticated features such as part-of-speech tags and dependency triples, it sometimes works better than n-gram features. When source website information is added, the n-grams features get better. Generally, source website information outperforms POS tag, and both of them work better than dependency triples. All those features are noisy and feature selection is needed to obtain better performance. For example, the embedding vectors ("W2V") does not work as well as others when the prediction is made by the linear regression model. When elastic net is employed, it can get comparable performance with other baseline methods. However, the overfitting problem cannot always be fully solved with L-1 and L-2 regularization.

The knowledge graph-based features do not always work better than the baseline features. The performance of lexical, syntactic and semantic features is generally comparable. Entity-based and explanatory-based features without context information are not very effective for the revenue prediction task, even worse than the baseline features. When context information

Table 3.3: Performance in prediction task with elastic net model

feature	Weekend revenue		Per-screen revenue	
	MAE($M)	SMAPE (%)	MAE($K)	SMAPE (%)
unigram	8.45	69.68	6.48	34.03
uni- and bigram	8.20	67.96	6.29	31.65
uni-, bi- and trigram	8.19	67.13	**6.15***	**30.96***
POS	**7.93**	67.91	6.92	34.81
DEP	8.33	67.42	6.46	32.84
W2V	8.19	**66.03**	6.44	32.34
Wiki(EF)	9.25	70.56	7.35	36.14
Yago(EF)	9.43	72.56	7.01	35.43
Wiki(CaEF)	9.10	70.54	6.97	34.84
Yago(CaEF)	8.85	69.19	6.68	33.64
Wiki(ExPF)	10.02	71.10	7.18	36.20
Yago(ExPF)	10.09	70.90	7.04	35.52
Wiki(CaExPF)	9.03	70.33	6.70	33.86
Yago(CaExPF)	8.86	68.24	6.50	33.19

Two-tailed t-test is done for paired data. In each pair, one is the method that get best performance, and the other one is any of the other methods. * indicates p-value < 0.01 and † indicates p-value < 0.05.

is brought in, more powerful features are generated (i.e., context-aware entity-based features denoted as CaEF and context-aware explanatory path-based features denoted as CaExPF). Among all the knowledge graph-based features, the context-aware explanatory path-based features are the best. However, the overall performance of all the knowledge graph-based features is not satisfactory. Maybe one reason is that the coverage of the original knowledge graph is limited so the effectiveness of the knowledge graph-based is impacted.

3.4.4 Summary

In general, the proposed knowledge graph-assistant featurization methods are beneficial in the entity-centric prediction applications, especially the classification task. The entity-based features are more effective than the explanatory path-based features in the classification task. When combining with context information, both the entity-based features and the explanatory path-based features are further enhanced. In the more general regression task which is to predict the revenue performance of movies, context-aware explanatory path-based features work the best. However, the revenue prediction task is much more challenging than the review classification task, and neither the baseline methods nor our methods are able to

make accurate prediction. One possible reason why the performance of the knowledge graph-based features is not outstanding especially for the regression task is that the coverage of the knowledge graph is not sufficient and many of the related entities or relations are missing in the knowledge graph. In the next chapter, we will testify this hypothesis and see whether the expansion of knowledge graph can improve the performance of knowledge graph-assistant featurization methods.

CHAPTER 4: EXPAND KNOWLEDGE GRAPH

4.1 INTRODUCTION

In general, knowledge graph is a powerful tool to facilitate deeper understanding of text. Recent years have seen much progress towards creating universal and domain knowledge graphs and many open sources of knowledge graph are available online. Some of them are in large scale or even web scale, which makes them offer significant benefit in a variety of applications. These knowledge graphs can help us to discover relations between entities, finally leading to a better capture of correlation between text and response variables in entity-centric tasks.

Relations covered by knowledge graph usually fall into predefined types, which allows for better organization of human knowledge. However, relation between words or entities can go beyond the scope of the covered relationships in an existing knowledge graph. For example, a book can be connected with its author and publisher in the knowledge graph. If we want to understand people's feedback about the book, such information could be helpful. On the other hand, the word "reader" may not be closely connected to the book entities in a knowledge, but its mention in the document could provide important signals about readers' attitude. Such relationships, though more implicit compared to the well-organized predefined relation types, are also useful for the entity-centric prediction tasks.

In practice, no matter how large a knowledge graph is, it is infeasible to cover all the overt and implicit word relations and entity relations. To solve this problem, we propose to mine word relations from the background text corpus and extend the knowledge graph by those implicit relations between words and entities.

There are two fundamental and complementary types of interesting semantic relations between words in natural languages. The first is the relation between two words that tend to *occur in similar context*; such a relation connects distributionally similar words. The second is the relation between two words that tend to *co-occur* with each other together; such a relation connects statistically associated words. In semiotics, the first type of relation is called *paradigmatic relation*, and the second *syntagmatic relation*.

To illustrate these two relationships, consider two synonyms such as "car" and "vehicle", which is a good example of words that have a paradigmatic relation because they tend to occur in the same context. If we substitute one for the other in a sentence, we would still have a meaningful sentence, whereas two semantically associated words such as "car" and "drive" would have a syntagmatic relation because they tend to co-occur in the same sentence (note

that we generally would not obtain a meaningful sentence by substituting "car" for "drive" or "drive" for "car"). Table 4.1 gives another example of syntagmatic and paradigmatic relation. "Paris" and "Chicago" are strongly related in terms of paradigmatic relation and so are "Monday" and "December", whereas "arrived" and "Chicago" are syntagmatically related, and so are "go" and "Paris".

In general, syntagmatic relation emphasizes positioning and co-occurrence in the same context, whereas paradigmatic relation usually holds between words that are associated with each other in the same category and can be substitutional in many contexts. Different from syntagmatic relation, paradigmatic relation does not require words to co-occur in the same context at the same time.

As two basic relationships, syntagmatic relation and paradigmatic relation are complementary with each other. Paradigmatic relation tells us how words are associated with one another as playing similar roles in terms of functional rule, thus often capturing synonym-like relations, while syntagmatic relation reveals how words can be combined with each other to complete the functional synthesis, thus often capturing topically associated words.

Both paradigmatic and syntagmatic relations are very useful knowledge fundamental to various applications involving text processing, including, e.g., search engines, recommender systems, text classification, text summarization, and text analytics. For example, such relations can be directly useful for query expansion in search engine applications to enrich the representation of a query or suggest related queries, and for capturing inexact matching of text for classification or clustering. Though they do not classify the relation between words or entities to the pre-defined categories as the way how a traditional knowledge graph organizes the relations, paradigmatic and syntagmatic relation can capture the association between two words in a higher level which can be used to find related entities.

Table 4.1: Example for syntagmatic relation and paradigmatic relation.

	Paradigmatic Relationship
Syntagmatic Relationship	She has arrived at Chicago on Monday
	I will go to Paris in December

Text data are unstructured, and effective discovery of knowledge from text data requires the computer to understand natural languages, which is known to be an extremely difficult task due to the dependency on other difficult tasks in artificial intelligence such as knowledge representation and reasoning. How to develop general algorithms to systematically discover semantic relations of words from arbitrary text data in a scalable way is a major open challenge in text data mining. In this chapter, we will make use of these two fundamental

types of word relations to expand knowledge graph to enlarge the coverage.

We will mainly focus on how to mine paradigmatic and syntagmatic relations from the background text corpus and make use of them to expand knowledge graph to achieve better featurization of text data for entity-centric applications. First, we propose several different methods to extract paradigmatic and syntagmatic relations from the text. Next, we will make use of the discovered paradigmatic and syntagmatic relations as a complement to an existing knowledge graph in the text featurization to check whether enlarging the coverage of knowledge is important. We evaluate our method in two applications: movie review classification and revenue prediction. The experimental results show that the proposed method generates better representation of documents thus improves the performance.

4.2 EXPAND KNOWLEDGE GRAPH BY PARADIGMATIC AND SYNTAGMATIC RELATIONSHIPS

4.2.1 Discovering paradigmatic and syntagmatic relationship by the random walk-based model

In this section, we study how to mine large text data in an unsupervised way to discover paradigmatic and syntagmatic relations efficiently and effectively [47]. We propose a novel general probabilistic approach based on random walks on word adjacency graphs to systematically mine these two fundamental and complementary lexical relations between words from arbitrary text data. In particular, we show that representing text data as an adjacency graph opens up many opportunities to define interesting random walks on the graph for mining lexical relation patterns, and propose multiple types of random walks for mining useful paradigmatic and syntagmatic relations. A word adjacency graph is a graph with words as vertices and edges indicating whether two words have adjacency relation in the text data. Edges have weights which can be computed based on word co-occurrences, i.e., the estimated probability that two words occur "next" to each other. Note that we define "next" very generally to include both immediate adjacency of two words occurring in text (i.e., one immediately after another) and sequential co-occurrence of the two words with gaps in between.

For example, given two sentences, "The vegetable grows fast in our greenhouse" and "The number of possible connection grows exponentially", if we only consider immediately adjacency, the adjacency graph would be as shown in Figure 4.1.

We can also generate other versions of adjacency graphs by making use of non-immediately adjacent co-occurrence, where gaps are allowed for co-occurring words. For example, if we

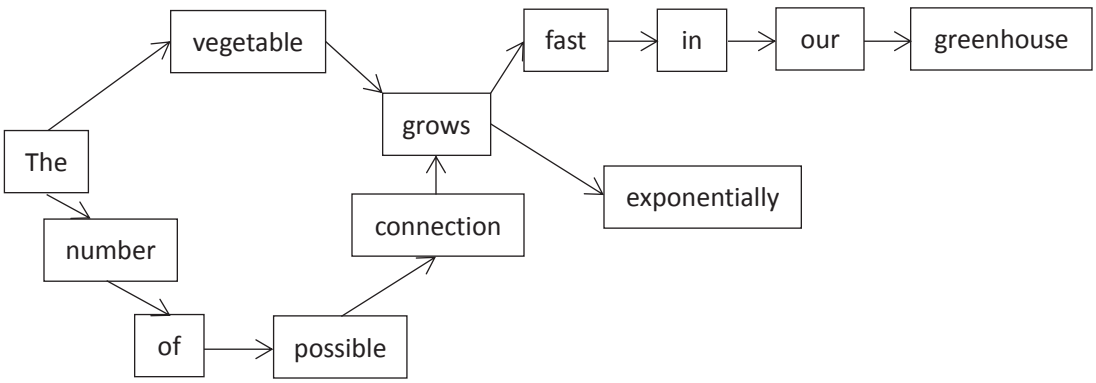


Figure 4.1: An example of word graph.

allow one gap between two adjacent words and add an edge between them, a part of the graph derived from the above example would look like "grown" → "in" → "greenhouse". Allowing non-immediate adjacency enables discovery of long-distance relations and better use of the corpus statistics.

The basic idea of the random walk based approach is to introduce a random walker that can stochastically "walk" on the nodes of the graph by following the edges, with higher probabilities of following edges that have higher weights (i.e., higher counts of co-occurrences). The probability of a trajectory of such a random walker can then be associated with the frequency of observing the word co-occurrences visited by the walker. A random walker can also walk backward along the edges of the graph (i.e., in the reverse direction of an edge), capturing different word co-occurrences than walking forward. Furthermore, we may also define "round trips" of the random walker where the walker would first go forward (or backward) and then backward (or forward).

With random walks defined on graphs, we can quantify the relation between two words based on the probability of the random walker walking from one word to reach the other in various ways. For example, words that tend to occur in similar context have high probabilities to be reached in random walks starting from one to the other in both forward and backward directions. Also, a round trip random walk going back to a word itself can be used to discover words associated with the starting word based on how likely it would go through a particular candidate word. Although not explored in this thesis, similar approaches can also be applied to mine interesting word sequences based on most probable paths of a random walk, which can be useful for text summarization or topic extraction. Thus using random walks on word graphs opens up an interesting new way to mine text data, especially for discovery of useful lexical associations.

Such a new approach has several important advantages: 1) It is completely general and

unsupervised, thus it requires no/little human effort and can be applied to mine arbitrary text data in any natural language. 2) It is a principled probabilistic approach with a solid foundation based on random processes, thus the scores to quantify lexical relations are meaningful, and it is easy to adapt the approach to capture different types of semantic relations between words by simply changing the way a random walk is defined. 3) It is very efficient for discovering paradigmatic and syntagmatic relations, and thus can potentially scale up to mine very big text data. It is generally easy to parallelize such algorithms since strongest associations tend to be "local" on the adjacency graph and multiple random walks can be potentially computed in parallel. Updating the graph is very efficient as we only need to update the co-occurrence statistics, making such a method scale up well to handle the "never-ending growth" of big text data in the real world in an online manner.

Although the general approach of random walks on word graphs can potentially solve many different text mining problems, we focus on systematically mining paradigmatic and syntagmatic relations. We propose and evaluate several random walk based algorithms for discovering such relations on real text data. Evaluation results show that the proposed algorithms are effective for mining these two kinds of relational patterns and can discover quite meaningful lexical knowledge from large text data without any human effort. Due to the generality and scalability, the proposed algorithms can be potentially applied to large amounts of arbitrary text data in different natural languages for discovery of useful lexical knowledge.

The random walk-based model is based on word adjacency graph which can be constructed from arbitrary text. Word is the basic element in text and a word adjacency graph is a graph with words as vertices and edges indicating whether two words have adjacency relation in the text data. Note that though we name it as word adjacency graph, it is not limited to only words, but the node can also be phrases or entities. Entities that can be recognized from the text are directly used as nodes in the word adjacency graph.

To construct a word adjacency graph, we connect every word or entity with its neighbors by a directed edge, and the direction is from the preceding one to the following one. The adjacency can be both immediate (i.e., one immediately after another) and non-immediate (i.e, co-occurrence with gaps in between). For example, given two sentences, "The vegetable grows fast in our greenhouse" and "The number of possible connection grows exponentially", if we only consider immediate adjacency, the adjacency graph would be as shown in Figure 4.1.

Formally, the node set in a word adjacency graph is denoted as $\mathcal{W}$ and the edge set is denoted as $\mathcal{E}_{\mathcal{W}}$. In $\mathcal{E}_{\mathcal{W}}$, an edge is represented by a triple $\langle w_i, w_j, c_{ij} \rangle$. Here w_i and w_j are the i-th and j node in $\mathcal{W}$ respectively and they are connected by the edge. c_{ij} is the number

of co-occurrence of w_i and w_j. Note that the co-occurrence is ordered, and $c_{ij} \neq c_{ji}$.

Two basic types of random walk can be defined on the word adjacency graph, namely forward walking and backward walking. Forward walking follows the direction of the edges while backward walking is in the inverse direction. Assume we have a path composed of an edge series (w_1, w_2), (w_2, w_3), ... ,(w_{n-1}, w_n) in the word adjacency graph where w_i is the i-th word (or entity) in this path. A forward walking $w_1 \rightarrow w_2 ... \rightarrow w_n$ is to visit w_1, w_2,..., w_n sequentially. And a backward walking $w_n \dashrightarrow w_l ... \dashrightarrow w_1$ is to visit w_n, w_l,..., w_1 by taking the inverse direction of edges between them.

Definition 4.1. An n-step forward walking $w_i \overset{n}{\rightarrow} w_j = \{w_i \rightarrow w_{r_1} \rightarrow w_{r_2} ... \rightarrow w_{r_{n-1}} \rightarrow w_j | (w_i, w_{r_1}), ..., (w_{r_{n-1}}, w_j) \in \mathcal{E}_{\mathcal{W}}\}$ and an n-step backward walking $w_i \overset{n}{\dashrightarrow} w_j = \{w_i \dashrightarrow w_{r_1} \dashrightarrow w_{r_2} ... \dashrightarrow w_{r_{n-1}} \dashrightarrow w_j | (w_{r_1}, w_i), ..., (w_j, w_{r_{n-1}}) \in \mathcal{E}_{\mathcal{W}}\}$.

For the example shown in Figure 4.1, "number"$\rightarrow$"of"$\rightarrow$"possible" $\rightarrow$"connection" is a 3-step forward walking and "connection"$\dashrightarrow$"possible" $\dashrightarrow$"of"$\dashrightarrow$"number" is a 3-step backward walking.

The probability of an n-step forward walking from w_i to w_j in a word adjacency graph is denoted as $P(w_i \overset{n}{\rightarrow} w_j)$ and the probability of an n-step backward walking is denoted as $P(w_i \overset{n}{\dashrightarrow} w_j)$, and they are defined as:

$$P(w_i \overset{n}{\rightarrow} w_j) = \sum_{w_r \in \mathcal{W}} P(w_i \overset{n-1}{\rightarrow} w_r) \cdot P(w_r \overset{1}{\rightarrow} w_j), \tag{4.1}$$

$$P(w_i \overset{n}{\dashrightarrow} w_j) = \sum_{w_r \in \mathcal{W}} P(w_i \overset{n-1}{\dashrightarrow} w_r) \cdot P(w_r \overset{1}{\dashrightarrow} w_j). \tag{4.2}$$

The probability of n step walking can be recursively derived from that of an 1-step walking. The probability of 1-step forward walking is

$$P(w_i \overset{1}{\rightarrow} w_j) = \frac{c_{ij}}{\sum_k c_{ik}}, \tag{4.3}$$

while the probability of 1-step backward walking is

$$P(w_i \overset{1}{\dashrightarrow} w_j) = \frac{c_{ij}}{\sum_k c_{kj}}. \tag{4.4}$$

Paradigmatic relation captures substitution and categorization of words. It usually holds between words that occur in similar context but not at the same time, which tend to share

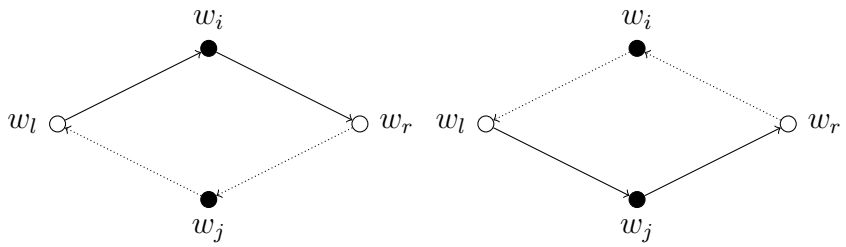


Figure 4.2: Illustration of circle trip for paradigmatic relation mining. (a) Clockwise circle trip; (b) Anti-clockwise circle trip.

common neighbors in the adjacency graph. If we take a random walk starting from w_i, visit the left and right neighbors and finally return to w_i, it is likely that we will pay a visit to w_j as it tends to be a popular connector among the neighbors. This observation motivates us to make use of "circle trip" over the context to quantify paradigmatic relation.

In this thesis, we take l-step circle trips to discover paradigmatic relationship, in both clockwise and anti-clockwise directions. In the first step, we need to find all common neighbors that are reachable from w_i and w_j by an l-step forward or backward walking in the adjacency graph.

As the example shown in Figure 4.2, w_l and w_r are common neighbors of w_i and w_j from left side and right side respectively, which satisfy the following conditions: $P_G(w_l \overset{l}{\rightarrow} w_i) > 0$, $P_G(w_l \overset{l}{\rightarrow} w_j) > 0$, $P_G(w_i \overset{l}{\rightarrow} w_r) > 0$ and $P_G(w_j \overset{l}{\rightarrow} w_r) > 0$. Then the clockwise circle trip between them is to walk in the order of $w_i \overset{l}{\rightarrow} w_r \overset{l}{\dashrightarrow} w_j \overset{l}{\dashrightarrow} w_l \overset{l}{\rightarrow} w_i$ (see Figure) and the anti-clockwise trip is $w_i \overset{l}{\dashrightarrow} w_l \overset{l}{\rightarrow} w_j \overset{l}{\rightarrow} w_r \overset{l}{\dashrightarrow} w_i$ (see Figure). To distinguish w_i and w_j from w_l and w_r, we say that w_i and w_j are vertical ends and w_l and w_r are horizontal ends.

Strong paradigmatic relation does not only require words to share enough context, but also emphasizes on their substitutability in the same context, which makes it easier for a random walker to accomplish both the clockwise and the anti-clockwise circle trip with w_i and w_j as the vertical ends. We denote the probability of taking l-step circle trip with vertical ends of v_i and v_j in both clockwise and anti-clockwise directions in adjacency graph G as $P_G(w_i \overset{l}{\circlearrowleft} w_j)$. If we denote $\{w_l | P_G(w_l \overset{l}{\rightarrow} w_i) > 0 \wedge P_G(w_l \overset{l}{\rightarrow} w_j) > 0\}$ as $\mathbb{L}$ and $\{w_r | P_G(w_i \overset{l}{\rightarrow} w_r) > 0 \wedge P_G(w_j \overset{l}{\rightarrow} w_r) > 0\}$ as $\mathbb{R}$, then

$$
\begin{aligned}
P_G(w_i \overset{l}{\circlearrowleft} w_j) = \sum_{w_l \in \mathbb{L}} \sum_{w_r \in \mathbb{R}} & P_G(w_i \overset{l}{\rightarrow} w_r) \cdot P_G(w_r \overset{l}{\dashrightarrow} w_j) \\
& \cdot P_G(w_j \overset{l}{\dashrightarrow} w_l) \cdot P_G(w_l \overset{l}{\rightarrow} w_i) \\
\cdot \sum_{w_l \in \mathbb{L}} \sum_{w_r \in \mathbb{R}} & P_G(w_i \overset{l}{\dashrightarrow} w_l) \cdot P_G(w_l \overset{l}{\rightarrow} w_j) \\
& \cdot P_G(w_j \overset{l}{\rightarrow} w_r) \cdot P_G(w_r \overset{l}{\dashrightarrow} w_i)
\end{aligned} \tag{4.5}
$$

However, it may cause bias towards frequent words if we only make use of $P_G(w_i \overset{l}{\circlearrowleft} w_j)$ to extract paradigmatic relation. When $|\mathbb{L}|$ and $|\mathbb{R}|$ are large enough, no matter whether w_i and w_j are substitutable in the contexts or not, the value of $P_G(w_i \overset{l}{\circlearrowleft} w_j)$ will be high. To tackle this problem, we normalize $P_G(w_i \overset{l}{\circlearrowleft} w_j)$ by the number of all possible circle trips in a unique direction, which is $\frac{1}{|\mathbb{L}| \cdot |\mathbb{R}|}$. The normalized $P_G(w_i \overset{l}{\circlearrowleft} w_j)$ reflects how w_i and w_j are correlated with their common context on average and a high value of it implies that they are likely to be substitutable in the contexts.

Given that the random walker can choose different l to complete the circle trip, we will have:

$$
\sum_{l=0}^{s} P_G(w_i \overset{l}{\circlearrowleft} w_j) \cdot \frac{1}{|\mathbb{L}| \cdot |\mathbb{R}|} \cdot \alpha_l, \tag{4.6}
$$

where s is the max step allowed in the circle trip. α_l is the prior probability for the random walker to chose l. If short-distance circle trips are more reliable, small l will get higher α_l. If different multi-step circle trips are equally favored regardless of the path length, α_l is set equivalently for different l. Principally, α_l should be non-negative and $\sum_{l=0}^{s} \alpha_l = 1$.

Combining different adjacency graphs induced from the same data together, we finally use $Pr(w_i, w_j)$ to measure the paradigmatic relation, which is defined as:

$$
Pr(w_i, w_j) = \sum_{k=1}^{K} \beta_k \sum_{l=0}^{s} P_{G_k}(w_i \overset{l}{\circlearrowleft} w_j) \cdot \frac{1}{|\mathbb{L}| \cdot |\mathbb{R}|} \cdot \alpha_l, \tag{4.7}
$$

where $\beta_k \geqslant 0$ and $\sum_{k=1}^{K} \beta_k = 1$.

Syntagmatic relation concerns adjacency and co-occurrence between words, and is usually sensitive to the order of words. This kind of correlation can be captured by round trips on the adjacency graph. If v_i and v_j co-occur a lot and v_i always locates before v_j, the probability of taking a forward trip from v_i to v_j and then walking back from v_j to v_i is likely to be high. The advantage of round trip over one-way trip is that round trip will

be less likely to be dominated by "popular" nodes which have a large number of outlinks or inlinks in the graph. Especially, "popular" nodes may bring in a lot of noise when the distribution of frequency of edges obey power law, which is followed by a wide variety of data including text data. In a round trip, even if it is quite easy for a random walker to reach a popular node, it is unlikely that the walker will easily return to the starting point from the popular node since there are too many paths to chose to walk back. Thus, we can explore syntagmatic relation based on two types of round trip random walk.

If the task for a random walker is to take an l-step forward walking from v_i to v_j and then return to v_i by an l-step backward walking, where l should be less than s and can be chosen in advance with a probability of α_l. Among all the possible round trips, the probability for the random walker to reach v_j as the destination is:

$$\begin{aligned} P_G^s(v_i \longrightarrow v_j) &= \frac{\sum_{l=1}^{s} P_G(v_i \overset{l}{\rightarrow} v_j) \cdot P_G(v_j \overset{l}{\dashrightarrow} v_i) \cdot \alpha_l}{\sum_{v_{j'} \in V} \sum_{l=1}^{s} P_G(v_i \overset{l}{\rightarrow} v_{j'}) \cdot P_G(v_{j'} \overset{l}{\dashrightarrow} v_i) \cdot \alpha_l} \\ &\propto \sum_{l=1}^{s} P_G(v_i \overset{l}{\rightarrow} v_j) \cdot P_G(v_j \overset{l}{\dashrightarrow} v_i) \cdot \alpha_l \end{aligned} \tag{4.8}$$

Here $\alpha \geqslant 0$ and $\sum_{l=1}^{s} \alpha_l = 1$.

We can define a similar task of backward-first round trip in which the first step is to take backward walking, and the probability of the random walker to successfully reach v_j is

$$\begin{aligned} P_G^s(v_i \longleftarrow v_j) &= \frac{\sum_{l=1}^{s} P_G(v_i \overset{l}{\dashrightarrow} v_j) \cdot P_G(v_j \overset{l}{\rightarrow} v_i) \cdot \alpha_l}{\sum_{v_{j'} \in V} \sum_{l=1}^{s} P_G(v_i \overset{l}{\dashrightarrow} v_{j'}) \cdot P_G(v_{j'} \overset{l}{\rightarrow} v_i) \cdot \alpha_l} \\ &\propto \sum_{l=1}^{s} P_G(v_i \overset{l}{\dashrightarrow} v_j) \cdot P_G(v_j \overset{l}{\rightarrow} v_i) \cdot \alpha_l \end{aligned} \tag{4.9}$$

If $P_G^s(v_i \longrightarrow v_j)$ is high, it indicates that v_j is likely to share strong syntagmatic relationship with v_j in the order of taking v_i as predecessor, whereas $P_G^s(v_i \longleftarrow v_j)$ measures their syntagmatic relatedness by taking v_i as a successor. For example, $P_G^s(v_i \longrightarrow v_j)$ can be high when v_i represents "more" and v_j represents "than", and $P_G^s(v_i \longleftarrow v_j)$ could be high if v_j is "much". If we further make use of multiple adjacency graphs, we can finally measure the syntagmatic relation in the following way:

$$Syn(v_i \rightarrow v_j) = \sum_{k=1}^{K} \beta_k \cdot P^s_{G_k}(v_i \longrightarrow v_j) \tag{4.10}$$

$$Syn(v_i \leftarrow v_j) = \sum_{k=1}^{K} \beta_k \cdot P^s_{G_k}(v_j \longleftarrow v_i) \tag{4.11}$$

where $Syn(v_i \rightarrow v_j)$ is the syntagmatic relation between v_i and v_j by taking the order of v_i being followed by v_j, while $Syn(v_i \leftarrow v_j)$ is in the inverse order.

Table 4.2, Table 4.3 and Table 4.4 show some examples of the paradigmatically related and syntagmatically related word pairs discovered from a set of news articles (TREC AP88 and AP89) by the random walk-based method respectively. From these examples, we can have a better understanding about what kind of related words that can be discovered by those two relationships in practice. For example, the weekday names are grouped together as the paradigmatically related words. While syntagmatic relation can tell us about interesting topics of the target word. For example, in the case of "Chinese", we can see the associated words show different aspects of topics, such as education, politics.

Table 4.2: Top 10 associated words retrieved by paradigmatic relation.

Monday	protein	more	year	intimacy
Tue	protein	more	week	intimacy
Thur	protien	less	year	contact
Mon	pellucida	stockier	month	intercourse
Thursday	renin	stouter	decade	encount
Wednesday	icam	shrewder	half-century	assault
Tuesday	insulin'secret	smoggier	day	relationship
Sunday	feedstuff	faser	century	tryst
Friday	DNA	worse	year-and-a-half	relate
Monday	gene-engine	dearer	quarter-century	liaison
Saturday	lecithin	faster	hour	abuse
can	slightly	eggplant	Berkeley	jump
can't	cent	eggplant	Berkeley	jump
can	slightly	celery	Livermore	rose
will	geffenplantinum	cantaloup	trucke	fell
could't	sharply	kale	Arcata	drop
would	broadly	jalapeno	Irvine	of
don't	modestly	honeydew	Riverside	increase
could	percent	melon	Cupertino	leap
cannot	issue	whiterib	Greenbrae	go
must	outnumber	onion	Sacramento	climb
should	mostly	watermelon	Lompoc	and

Table 4.3: Top 10 associated words retrieved by syntagmatic relation (top words retrieved from the right side).

Chinese →	school →	express→	long→	mount →
student	district	concern	island	Rushmore
mainland	superintendant	regret	overdue	Everest
leader	dropout	satisfaction	time	Hermon
government	student	dismay	beach	Rainier
opera	board	optimism	distance	Pleasant
author	teacher	wieczorny	way	Vernon
dissident	diploma	gratitude	enough	Holyoke
cheongsam	principal	sympathy	enough	Clemen
riben	bus	confidence	haul	Kisco
embassy	gymnasium	displeasure	ago	Tokachi

Table 4.4: Top 10 associated words retrieved by syntagmatic relation (top words retrieved from the left side).

→ information	→ market	→ spokesman	→ war	→ own
classify	over-the-count	ministry	world	their
non-public	stock	police	civil	his
confidential	bond	department	Iran-Iraq	wholly
inside	exchange	house	Vietnam	its
withhold	the	a	star	our
mcgrawhill	financial	embassy	postworld	my
IDD	broader	white	cold	her
sensitive	credit	army	eight-year	your
provide	secondary	pentagon	Korean	already
gather	BcCredit	FAA	beanfield	jointly

4.2.2 Discovering paradigmatic and syntagmatic relationship by the word embedding-based model

The random walk-based method directly uses the occurrence information between words to calculate the paradigmatic or syntagmatic similarity between words. An alternative way is to represent the words in a latent semantic space and then compute the paradigmatic and syntagmatic similarity based on the vectors. Word embedding technique is a well-studied and widely-used way to solve this problem. In this thesis, we use Skip-gram [48] to learn the embedding vectors. Skip-gram model can learn an input vector and an output vector for

Algorithm 4.1: Extension of knowledge graph

Input: Knowledge graph G, text corpus D, similarity measurement function $\mathcal{S}$
Output: Extended knowledge graph G'

1 **repeat**
2 $\quad e \leftarrow \text{argmax}_{e \notin E}\{\max_{e'}\{\mathcal{S}(e, e')|e' \in E\}\}$;
3 $\quad G' \leftarrow G' \cup \{(e, R, e')|e' \in \text{argmax}_{\hat{(e)}}\{\mathcal{S}(e, \hat{(e)})|\hat{(e)} \in E\})\}$;
4 $\quad iter \leftarrow iter + 1$;
5 **until** $iter > K$;

each word at the same time. The cosine similarity of the input vectors are usually used to represent words and the cosine similarity between them measures the paradigmatic relations between words. With both the input and output vectors, we can calculate the probability of seeing a word in the context of another one, which is a natural way to measure the syntagmatic similarity. Denote the input vector of word w as v_w and its output vector as v'_w, then the syntagmatic relation between w_i and w_j are calculated as:

$$\begin{aligned} Syn_{sg}(w_i, w_j) &= P(w_i|w_j)P(w_j|w_i) \\ &= \frac{exp(v'^T_{w_i} v_{w_j})}{\sum_{w_r} exp(v'^T_{w_r} v_{w_j})} \frac{exp(v'^T_{w_j} v_{w_i})}{\sum_{w_r} exp(v'^T_{w_r} v_{w_i})} \end{aligned} \tag{4.12}$$

The syntagmatic relation can be evaluated based on both input and output vectors, while the paradigmatic relation can simply measured by the cosine similarity between the input vectors, which is a commonly-used way to calculate word similarity:

$$Pr(w_i, w_j) = cos(v_{w_i}, v_{w_j}) \tag{4.13}$$

4.2.3 Expand knowledge graph

To expand the knowledge graph by paradigmatic or syntagmatic relation, we first extract all the entities mentioned in a document by NLP tools and then segment each sentence into phrases that represent the entity names and other words that are not recognized as entities. Then we calculate the similarity between each entity pair based on either of the above two methods. Finally, entities that are related to any of the existing entities in the knowledge graph are added into the knowledge graph. The procedure is summarized in Algorithm 4.1.

In Line 2, we examine all the entities that are not in the knowledge graph and calculate

their paradigmatic or syntagmatic similarity scores to the entities that are already contained in the knowledge graph. In this way, we compute similarity scores between each entity pairs where one is a new entity and the other is an existing entity in the knowledge graph. Then shown in Line 3 and Line 3, the corresponding new entity which gets the highest similarity score among all these pairs are added into the knowledge graph, along with its relation to the existing entity in the knowledge graph (the other entity in the pair). An existing entity is linked to the newly added one only if its similarity with the newly added one is larger than others. The relationship between them (R) is determined by the way how we measure the similarity, i.e., either paradigmatic or syntagmatic. Finally, the top new K entities which are most related to the existing ones in the knowledge graph are used to expand the knowledge graph.

4.3 EXPERIMENT

In this section, we apply our method in two applications: (1) classifying reviews according to the mentioned movies; (2) predicting the revenue of movies based on reviews. The experiment setup is the same as that in chapter 3. The window size of skip-gram is set to be 5, the step size in the random walk model is set to be 2. Two gap sizes are used to generate the word graph in the random walk model: 1 and 2. In the first word graph, only adjacent words are connected with each other. When the gap size is 2, one gap is allowed between words.

4.3.1 Knowledge graph expansion for the entity-centric classification task

We now confirm that bringing background knowledge graph from a knowledge graph is beneficial for entity-centric classification application. The next question is that whether extending the knowledge graph by mining word relations from local context can further improve the performance. As discussed in Section 4.2, two fundamental types of word relations – paradigmatic relation and syntagmatic relation – are used to discover related entities to extend the knowledge graph by either word embedding technique (Skip-gram) or a random-walk based method. When we get the extended knowledge graph, all the four methods of knowledge graph-based featurization is employed to represent the documents.

Similar to Chapter 3, we compare the knowledge graph-based features to several different types of baseline featurization methods, including lexical features, syntactic features such as POS tags and dependency triples, semantic features such as extracted named entities and

Table 4.5: Performance of extension of knowledge graph for entity-based featurization in classification task

feature	accuracy	precision	recall	F-1 measure
unigram	0.374	0.371	0.362	0.366
uni- and bigram	0.378	0.379	0.365	0.372
uni-, bi- and trigram	0.380	0.376	0.367	0.371
POS	0.270	0.249	0.243	0.246
DEP	0.171	0.158	0.152	0.155
NER	0.327	0.448	0.323	0.376
W2V	0.193	0.181	0.177	0.179
Wiki (EF)	0.339	0.419	0.332	0.370
Yago (EF)	0.303	0.439	0.296	0.353
Wiki + $Para_{rw}$ (EF)	0.440	0.548	0.430	0.482
Wiki + Syn_{rw} (EF)	**0.443**	0.551	**0.434**	0.486
Wiki + $Para_{sg}$ (EF)	**0.443**	**0.555***	**0.434**	**0.487**†
Wiki + Syn_{sg} (EF)	**0.443**	0.551	0.433	0.485
Yago + $Para_{rw}$ (EF)	0.439	0.541	0.428	0.478
Yago + Syn_{rw} (EF)	**0.443**	0.547	0.433	0.484
Yago + $Para_{sg}$ (EF)	0.430	0.543	0.420	0.481
Yago+ Syn_{sg} (EF)	0.442	0.543	0.431	0.473

Two-tailed t-test is done for paired data. In each pair, one is the method that get best performance, and the other one is any of the other methods. * indicates p-value < 0.01 for all tests and † indicates p-value < 0.05.

word embedding vectors. The original knowledge graph as well as the expanded one are both used to construct the knowledge graph-based features.

Table 4.5 shows the performance of expanded knowledge graph using entity-based featurization in classification task. The knowledge graph which is generated from Wikipedia Infobox is denoted as "Wiki". "Wiki + $Para_{sg}$" means to expand the knowledge graph by paradigmatic relation mined using Skip-gram model, while "Wiki + Syn_{rw}" represents the expansion based on syntagmatic relation discovered by the random walk-based method. "EF" means to construct the entity-based features using the corresponding knowledge graph, and "CaEF" denotes the context-aware entity-based features. We can see that because the coverage of the knowledge graph is significantly enlarged by adding related entities, the performance is significantly improved. Both paradigmatic relation and syntagmatic are effective in enlarging the coverage by either of the two methods. Because the original knowledge graph which is generated from Wikipedia Infobox and the meta data has a better coverage of the central entities as well as the related entities compared to Yago, it still works slightly better

Table 4.6: Performance of extension of knowledge graph for context-aware entity-based featurization in classification task

feature	accuracy	precision	recall	F-1 measure
unigram	0.374	0.371	0.362	0.366
uni- and bigram	0.378	0.379	0.365	0.372
uni-, bi- and trigram	0.380	0.376	0.367	0.371
POS	0.270	0.249	0.243	0.246
DEP	0.171	0.158	0.152	0.155
NER	0.327	0.448	0.323	0.376
W2V	0.193	0.181	0.177	0.179
Wiki (CaEF)	0.532	0.511	0.513	0.512
Yago (CaEF)	0.451	0.437	0.432	0.435
Wiki + $Para_{rw}$ (CaEF)	0.537	0.529	0.522	0.525
Wiki + Syn_{rw} (CaEF)	0.541	0.536	0.527	0.531
Wiki + $Para_{sg}$ (CaEF)	**0.555***	**0.549***	**0.543***	**0.546***
Wiki + Syn_{sg} (CaEF)	0.533	0.526	0.519	0.522
Yago + $Para_{rw}$ (CaEF)	0.482	0.483	0.467	0.475
Yago + Syn_{rw} (CaEF)	0.486	0.489	0.472	0.480
Yago + $Para_{sg}$ (CaEF)	0.493	0.495	0.480	0.487
Yago+ Syn_{sg} (CaEF)	0.479	0.480	0.465	0.472

Two-tailed t-test is done for paired data. In each pair, one is the method that get best performance, and the other one is any of the other methods. * indicates p-value < 0.01 for all tests.

after expansion. However, the expansion reduces the coverage gap between them and their performances are comparable. Generally, paradigmatic relation and syntagmatic relation achieves similar performance, and both are effective in finding related entities. The random walk-based methods is slightly more effective in finding syntagmatically related pairs while the paradigmatic relation discovered by skip-gram is more helpful in this task.

Context-aware entity-based features are also extracted from the text based on the expanded knowledge graphs, as shown in Table 4.8. We can arrive at the same conclusion as what we have for the entity-based features. Besides, similar to the evaluation results in Section 3.4, we can observe that the context-aware entity-based features work significantly better than the entity-based features, which again confirms that context information is of great value in the entity-centric classification task.

Table 4.7 demonstrates the comparison between the expanded knowledge graph and the original one using explanatory path-based features. We can see that with the help of expansion, the knowledge graph becomes much more powerful. It outperforms the baseline

Table 4.7: Performance of extension of knowledge graph for explanatory path-based featurization in classification task

feature	accuracy	precision	recall	F-1 measure
unigram	0.374	0.371	0.362	0.366
uni- and bigram	0.378	0.379	0.365	0.372
uni-, bi- and trigram	0.380	0.376	0.367	0.371
POS	0.270	0.249	0.243	0.246
DEP	0.171	0.158	0.152	0.155
NER	0.327	0.448	0.323	0.376
W2V	0.193	0.181	0.177	0.179
Wiki (ExPF)	0.230	0.364	0.226	0.278
Yago (ExPF)	0.217	0.348	0.212	0.263
Wiki + $Para_{rw}$ (ExPF)	0.370	0.471	0.362	0.409
Wiki + Syn_{rw} (ExPF)	0.379	0.455	0.365	0.405
Wiki + $Para_{sg}$ (ExPF)	**0.467***	**0.574***	**0.458***	**0.510***
Wiki + Syn_{sg} (ExPF)	0.411	0.517	0.404	0.453
Yago + $Para_{rw}$ (ExPF)	0.310	0.427	0.303	0.355
Yago + Syn_{rw} (ExPF)	0.368	0.450	0.355	0.397
Yago + $Para_{sg}$ (ExPF)	0.442	0.557	0.434	0.488
Yago+ Syn_{sg} (ExPF)	0.369	0.471	0.362	0.409

Two-tailed t-test is done for paired data. In each pair, one is the method that get best performance, and the other one is any of the other methods. * indicates p-value < 0.01 for all tests.

methods even though the explanatory path-based features are shown to be not effective when using the original knowledge graph, which indicates that the coverage of the knowledge graph is indeed of great importance.

Similarly, by combining the context information with the explanatory path, the constructed features can be further enhanced. Context-aware explanatory path-based featurization method is applied in Table 4.8. We can see that it beats the explanatory path-based features on all metrics.

We have a parameter K in Algorithm 4.1 to control the number of new entities that are added to the knowledge graph. Larger K results in larger coverage, but meanwhile may also increase the risk of bringing in more noise. To see how the setting of K will impact the performance of the expanded knowledge graph in the classification task, we construct different versions of the knowledge graph.

Based on the original knowledge graph constructed from Wiki Infobox and the meta data, we build multiple versions of expanded knowledge graph. Entity-based features are used.

Table 4.8: Performance of extension of knowledge graph for context-aware explanatory path-based featurization in classification task

feature	accuracy	precision	recall	F-1 measure
unigram	0.374	0.371	0.362	0.366
uni- and bigram	0.378	0.379	0.365	0.372
uni-, bi- and trigram	0.380	0.376	0.367	0.371
POS	0.270	0.249	0.243	0.246
DEP	0.171	0.158	0.152	0.155
NER	0.327	0.448	0.323	0.376
W2V	0.193	0.181	0.177	0.179
Wiki (CaExPF)	0.458	0.451	0.441	0.446
Yago (CaExPF)	0.307	0.311	0.291	0.301
Wiki + $Para_{rw}$ (CaExPF)	0.433	0.429	0.418	0.423
Wiki + Syn_{rw} (CaExPF)	0.452	0.428	0.432	0.430
Wiki + $Para_{sg}$ (CaExPF)	**0.557***	**0.548***	**0.544***	**0.546***
Wiki + Syn_{sg} (CaExPF)	0.480	0.473	0.468	0.471
Yago + $Para_{rw}$ (CaExPF)	0.306	0.315	0.294	0.304
Yago + Syn_{rw} (CaExPF)	0.379	0.367	0.360	0.363
Yago + $Para_{sg}$ (CaExPF)	0.450	0.453	0.437	0.445
Yago+ Syn_{sg} (CaExPF)	0.375	0.374	0.361	0.367

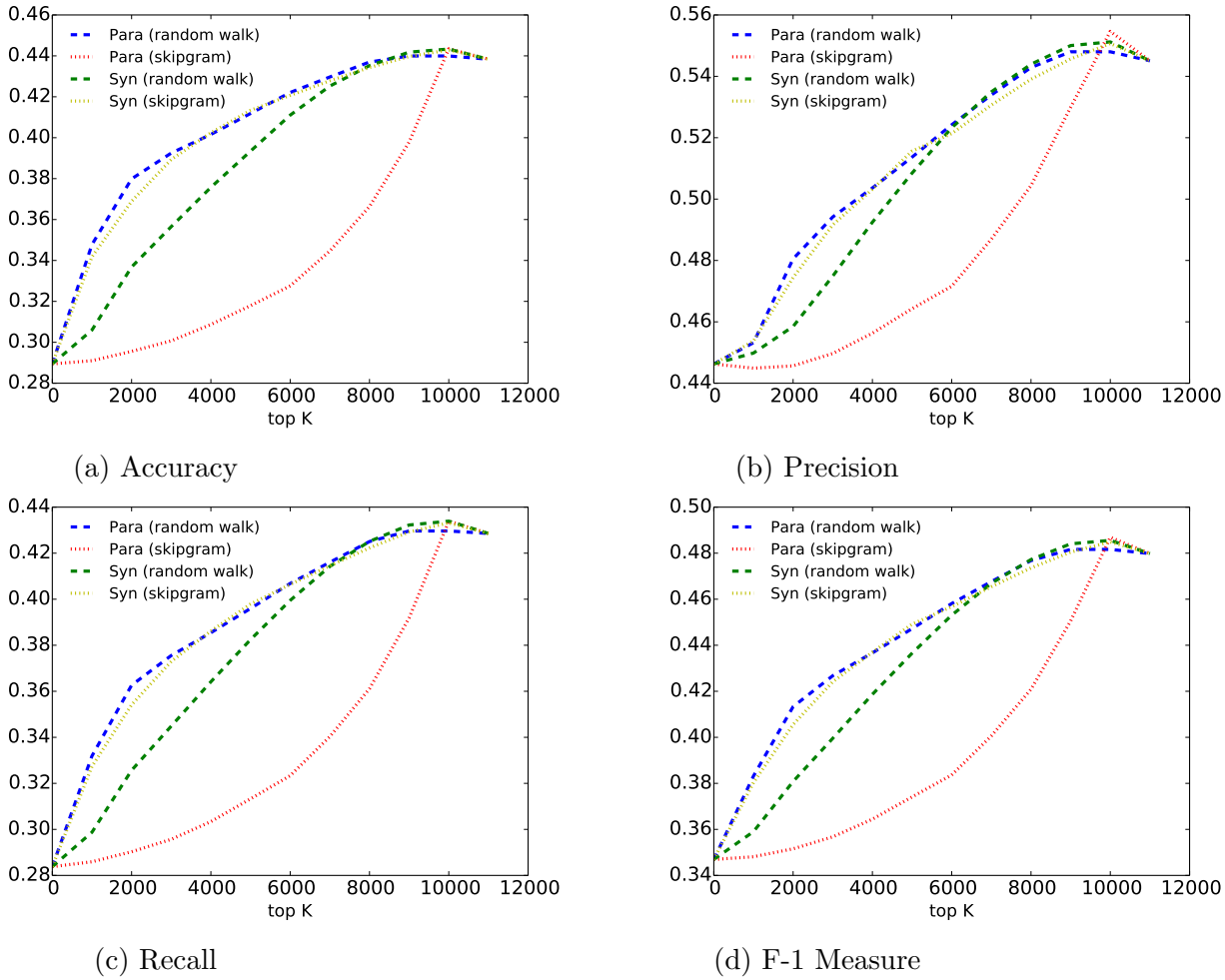
Two-tailed t-test is done for paired data. In each pair, one is the method that get best performance, and the other one is any of the other methods. * indicates p-value < 0.01 for all tests.

Their performances are shown in Figure 4.3. We can see that the top ranked pairs found by paradigmatic relation based on the random walk-based method are more reliable compared to others and the curve rises much faster at the beginning. With more and more entities added into the knowledge graph, all the curves arrive at similar value.

We also conduct the same experiments on Yago and the results are shown in Figure 4.4, from which we can draw similar conclusion. In general, with more entities are added into the knowledge graph, the performance gets better. However, the knowledge graph-based features will become noisy when too many entities are included, thus hurts the performance.

4.3.2 Knowledge graph expansion for the entity-centric regression task

In the Section 4.3.1, we show that leverage knowledge graph can provide tangible benefits to entity-centric classification tasks. A further question is that whether the knowledge graph-based features are helpful in entity-centric prediction tasks. Entity-centric prediction can be sometimes much more challenging than the classification task because the response variable



(a) Accuracy

(b) Precision

(c) Recall

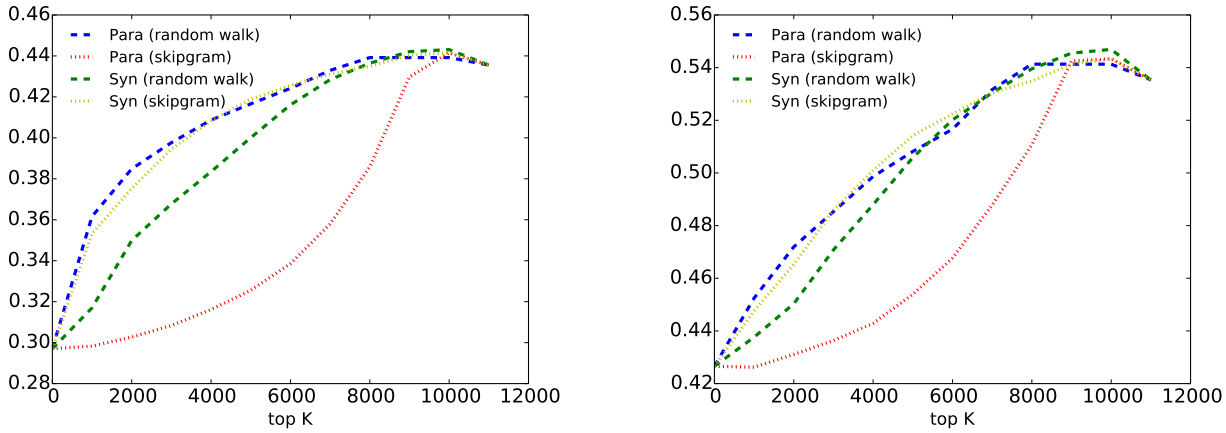
(d) F-1 Measure

Figure 4.3: Impact of K in knowledge graph expansion for the entity-centric classification task, with original knowledge graph constructed from Wiki Infobox and the meta data.

that is to be predicted is weakly related to the text information. We conduct the experiment to predict revenue performance based on movie reviews. The experiment includes two kinds of response variables, one is the weekend revenue and the other is per-screen revenue. There are two models used to make the prediction. "LR" represents the linear regression model and "Elastic net" is the linear regression model with L-1 and L-2 regularization.

When we only use the original knowledge graph to generate the features, it does not help much and the prediction errors are even larger than the baseline methods, as shown in Table 3.2 and Table 3.3. We suspect that the reason for the underperformance of the knowledge graph-based features is the insufficient coverage of the knowledge graph. In this section, we will take a close look into this problem.

As shown in Table 3.2 and Table 3.3, the context-aware explanatory path-based features (CaExPF) works best among all the knowledge graph-based features. Thus, we mainly focus



(a) Accuracy

(b) Precision

(c) Recall

(d) F-1 Measure

Figure 4.4: Impact of K in knowledge graph expansion for the entity-centric classification task, with original knowledge graph as Yago.

on using the context-aware explanatory path-based features to represent the documents, based on which we can further analyze the influence of knowledge graph expansion in the regression task.

We expand the knowledge graph by either paradigmatic or syntagmatic relations, just as what we did for the classification task. The results can be found in Table 4.9 and Table 4.10. Compared to the original knowledge graph, the expanded version works much better and achieve comparable prediction accuracy as the baseline methods, for both linear regression and elastic net model. If we combine the context-aware explanatory path-based features with lexical features such as unigram, the prediction error can be further reduced. However, the significance test shows that the reduction is not significant.

Though knowledge graph may help in some cases, how to make use of knowledge graph in such kind of applications where the text only provides weak signals for the prediction of the

Table 4.9: Performance of extension of knowledge graph with linear regression model in prediction task

feature	Weekend revenue		Per-screen revenue	
	MAE($M)	SMAPE (%)	MAE($K)	SMAPE (%)
unigram	9.41	70.71	7.07	38.13
uni- and bigram	9.03	68.78	6.70	33.61
uni-, bi- and trigram	9.05	68.55	6.68	33.50
POS	9.11	70.04	7.12	37.66
DEP	9.40	69.27	6.91	35.17
W2V	10.04	68.55	8.86	46.83
Wiki	10.83	69.93	7.18	36.49
Yago	10.43	73.06	6.90	35.29
Wiki+$Para_{rw}$	9.84	70.40	6.80	35.24
Wiki+$Para_{sg}$	9.84	70.89	6.90	35.55
Wiki+Syn_{rw}	9.89	69.00	7.03	36.76
Wiki+Syn_{sg}	9.82	71.40	6.66	34.51
Yago+$Para_{rw}$	9.73	69.44	6.94	35.78
Yago+$Para_{sg}$	9.62	69.39	6.91	35.35
Yago+Syn_{rw}	9.63	67.70	6.82	34.67
Yago+Syn_{sg}	9.56	69.56	7.07	35.85
Wiki+$Para_{rw}$ + unigram	8.94	69.40	**6.53**	**33.16**
Wiki+$Para_{sg}$ + unigram	**8.91**	**68.35**	6.67	33.89
Wiki+Syn_{rw} + unigram	8.99	68.53	6.79	34.67
Wiki+Syn_{sg} + unigram	9.00	69.38	6.51	33.30
Yago+$Para_{rw}$ + unigram	9.09	69.11	6.72	34.30
Yago+$Para_{sg}$ + unigram	9.03	69.91	6.66	34.07
Yago+Syn_{rw} + unigram	9.10	69.02	6.69	34.09
Yago+Syn_{sg} + unigram	8.97	70.21	6.75	34.29

Context-aware explanatory-path featurization (CaExPF) is used to represent the documents. Two-tailed t-test is done for paired data. In each pair, one is the method that get best performance, and the other one is any of the other methods. † indicates p-value < 0.05.

response variables is still very challenging problem. This again emphasizes the necessity of constructing a task-aware knowledge graph, generating best-suited knowledge graph-based features and selecting the features in a task-dependent way, especially when the correlation between the text and the response is weak to capture.

Table 4.10: Performance of extension of knowledge graph with elastic net model in prediction task

feature	Weekend revenue		Per-screen revenue	
	MAE($M)	SMAPE (%)	MAE($K)	SMAPE (%)
unigram	8.45	69.68	6.48	34.03
uni- and bigram	8.20	67.96	6.29	31.65
uni-, bi- and trigram	8.19	67.13	**6.15**	**30.96**
POS	7.93	67.91	6.92	34.81
DEP	8.33	67.42	6.46	32.84
W2V	8.19	66.03	6.44	32.34
Wiki	9.03	70.33	6.70	33.86
Yago	8.86	68.24	6.50	33.19
Wiki+$Para_{rw}$	8.44	68.49	6.70	34.36
Wiki+$Para_{sg}$	8.44	67.51	6.45	33.23
Wiki+Syn_{rw}	8.66	68.53	6.72	34.67
Wiki+Syn_{sg}	8.60	68.77	6.34	32.22
Yago+$Para_{rw}$	8.45	68.11	6.48	33.49
Yago+$Para_{sg}$	8.26	66.53	6.55	33.41
Yago+Syn_{rw}	8.43	66.96	6.37	32.63
Yago+Syn_{sg}	8.27	67.54	6.80	34.36
Wiki+$Para_{rw}$ + unigram	7.79	65.86	6.31	32.17
Wiki+$Para_{sg}$ + unigram	7.92	66.01	6.35	32.13
Wiki+Syn_{rw} + unigram	8.01	65.97	6.35	32.19
Wiki+Syn_{sg} + unigram	7.95	**65.68**	6.28	32.12
Yago+$Para_{rw}$ + unigram	8.05	66.73	6.31	32.11
Yago+$Para_{sg}$ + unigram	7.90	66.80	6.37	32.16
Yago+Syn_{rw} + unigram	7.99	66.33	6.36	32.23
Yago+Syn_{sg} + unigram	**7.77**	66.51	6.46	32.60

Context-aware explanatory-path featurization (CaExPF) is used to represent the documents. Two-tailed t-test is done for paired data. In each pair, one is the method that get best performance, and the other one is any of the other methods. † indicates p-value < 0.05.

4.3.3 Summary

In this section, we conduct experiment to examine the efficacy of expanding knowledge graph in the knowledge graph-assitant feature engineering. It turns out both the classification task and the regression task benefit from the expansion of the knowledge graph. The coverage of a pre-constructed knowledge graph is sometimes not sufficient for real-world applictions, which will eventually impact the effectiveness of using such a knowledge graph in the applications. When the major problem is limited coverage, enlarging the coverage is

more important than reducing the noise, thus a larger K (in Algorithm 4.1) to allow more entities to be added in the knowledge graph is favored. However, considerable noise will be brought in when K is too large. To avoid introducing substantial noise, it is necessary to control the quality of the newly added entities.

Compared to the classification task, though the expansion of the knowledge graph helps the knowledge graph-based features to work much better, the predictive power is still not good enough. Context-aware explanatory path-based features are shown to be most effective among all the four types of knowledge graph-based features, which requires the explanatory path to be accurate to support the utility of the features in the regression task. Although paradigmatic and syntagmatic can help to make significant enlargement of the related entities, they do not distinguish the precise relationship between the new entities and the central entities in a finer granularity. Hence, we argue that a more customized expansion of the knowledge graph is needed.

On one hand, the expansion should only focus on finding relevant entities or relations that are useful for the task in order to avoid bringing in noise. On the other hand, more precise relationship in a finer granularity needs to be recognized when new entities are introduced. For example, we may only want to include entities that are related to the existing entities in the knowledge graph by the pre-defined relationship. In this way, the explanatory path will be more accurate and thus can provide better explanation of the relatedness between the new entities and the central entities. In the next chapter, we will discuss the problem of how to construct such a knowledge graph.

CHAPTER 5: A FRAMEWORK FOR TASK-AWARE KNOWLEDGE GRAPH

As discussed in Chapter 3, entity-centric prediction can benefit from leveraging knowledge graph. However, a main factor that constrains the full exploitation of knowledge graph in such tasks is the limited coverage of the pre-constructed knowledge graph. When the construction and the application of knowledge graph are disjoint from each other, a knowledge graph will remain unchanged during its use. It receives no feedback from the task and thus cannot assimilate new useful knowledge from the text data.

We argue that it is important to expand the knowledge graph based on the background text corpus to enlarge the coverage. Experimental results in Chapter 4 have shown that the knowledge graph-based features can be more effective when we expand the knowledge graph by paradigmatic and syntagmatic relations. However, one problem with such kind of expansion is that paradigmatic and syntagmatic relations, though good at finding related entities, are not discriminative enough when we need to distinguish the finer-granularity relationships, which play an important role in constructing the context-aware explanatory path-based features.

To build a customized knowledge graph for a given task, we propose to construct a task-aware knowledge graph (TAKG) which would only contain the relevant knowledge to a particular task and assimilate new knowledge from the background text data. In this way, task-aware knowledge graph can provide better knowledge support for the particular task, by enlarging the coverage and reducing the noise. Meanwhile, it can identify the newly found relations more precisely compared to loosely connecting the new entities to the existing entities by paradigmatic and syntagmatic relation, which helps to get higher-quality explanatory path.

As an example, consider the task to predict the movie revenue performance based on the reviews. Part of a general knowledge graph that can be used to solve this problem is shown in Figure 1.2a which includes movie entities and their relations with other entities. If we find that whether the audience like the songs in the movie will have impact on the revenue. There is a review "City of Stars is my favorite song in La La Land", indicating that "City of Stars" is a song of "La La Land" which is unknown to the knowledge graph. Then we can expand the TAKG with this new knowledge. With paradigmatic and syntagmatic relation-based expansion, we may only be able to know that "City of Stars" is related to "La La Land", but cannot tell how it is different from the relation between "Emma Stone" and "La La Land". With a task-aware knowledge graph, we hope to learn about more detailed information like whether the relation between "City of Stars" and "La La Land" can fall

into some pre-defined categories such as "hasSong" or "starredBy". Besides discovering new knowledge from the text corpus, we also learn from the feedback of the task that relations such as "producedBy" is not very useful for the task, so we will just remove such kind of relations from the knowledge graph.

As shown in the above example, the construction of a TAKG includes (1) trimming the knowledge graph to reduce noise and increase efficiency and (2) expanding it with new knowledge that can be extracted from the data used in the task to enlarge coverage. The process can be repeated multiple times to iteratively adapt the knowledge graph to the task. TAKG bridges the gap between the construction and application of knowledge graph to allow for a potentially iterative process for constructing an increasingly relevant and complete knowledge graph customized toward maximizing the performance of a specific task.

5.1 DEFINITION OF TASK-AWARE KNOWLEDGE GRAPH

As discussed in Section 3.2, a knowledge graph consists of entities and relations between them. A task-aware knowledge graph also has a graph structure where nodes represent entities and edges represent relations. Besides, each entity and relation are both assigned with a task awareness value to evaluate their relatedness or usefulness for the task.

A task-aware knowledge graph can be formally defined as $G = \{E, R, t, \Sigma\}$ where E and R are again the entity set and relation set, and t is the task. Σ is the assignment of the task awareness, which evaluates all the entities, relations and attributes from the perspective of the relevance to the task. f is the function that evaluates the task awareness. Function f always returns a non-negative value, and the number can be binary (1 or 0) or a continuous value, with a zero indicating that the entity or relation is not relevant or useful for the task.

Definition 5.1. ***Task-aware knowledge graph:*** A task-aware knowledge graph can be represented as $G = \{E, R, t, \Sigma\}$ where E and R are the entity set and relation set respectively. Σ is the assignement of taskawareness measurement to each entity, relation and attribute in the knowledge graph: $\Sigma = \{(e, f(e,t))|e \in E\} \cup \{(r, f(r,t))|r \in R\} \cup \{(a, f(a,t))|a \in e \wedge e \in E\} \cup \{(a, f(a,t))|a \in r.A \wedge r \in R\}$.

Example 5.1. The knowledge graph in Figure 3.1 contains movies entities such as $e_1 = \{\langle name,$

$LaLaLand\rangle, \langle Budget, \$30million\rangle, ...\}$, which are connected with other types of entities with various relations. For example, the movie entity e_1 are connected two person entities: (1) its director by relations such as $\langle e_1, \langle type, DirectedBy\rangle, e_2\rangle$; (2) and the actress who starred in the movie $\langle e_1, \langle type, StarredBy\rangle, e_3\rangle$. Then the entity set of the knowledge graph is $\{e_1, e_2, \ldots\}$, while the relation sets is $\{\langle e_1, \langle type, DirectedBy\rangle, e_2\rangle, \langle e_1, \langle type, ProducedBy\rangle, e_3\rangle, \ldots\}$. From the feedback of the task which is to classify reviews according to the movies being reviewed, we learned that relations such as "*ProducedBy*" is not very useful for our task, then we just set its task awareness value as 0. The task awareness measurement assignment is $\Sigma = \{(e_1, 1), (\langle e_1, \langle type, ProducedBy\rangle, e_3\rangle, 0), \ldots\}$.

A task-aware knowledge graph is always constructed to help a particular task. In some sense, the task awareness measurement of entities and relations provides a "guideline" for the application of the knowledge graph in the task. Entities and relations with low task awareness value are unlikely useful for the task and can thus be ignored to reduce noise. We can also make use of the task awareness measurement in the construction of the task-aware knowledge graph. For example, we can prune the task-aware knowledge graph by removing entities and relations whose task awareness is zero. Expansion can also be made by finding new entities or relations that are similar or related to those with high task awareness.

We can see that a key component in a task-aware knowledge graph is the task awareness measurement function, which will be introduced in the next section.

5.2 TASK AWARENESS MEASUREMENT

A task awareness measurement function is designed to measure the relevance of an entity, relation or attribute to a task. All kinds of information provided by a general knowledge graph are valuable from the perspective of accumulating machine-usable human knowledge. However, it may not be necessary to include all of them in a particular task. Only the entities or relations that are relevant to the task need to be taken into consideration. For example, when we want to find people's opinion about a newly released product, the background knowledge we need can be all included in a small sub-graph composed of the product and some of its neighbor entities. Focusing on relevant entities and relations not only helps to improve efficiency, but also removes potentially distracting noise.

An entity, relation or attribute has high task awareness only if it is relevant to the task. The relevance can be judged by prior knowledge. For example, we can assume entities that are connected to the central entity by some types of relations are relevant to the task of mining opinions about products, and use the assumption as the prior knowledge. In this case, task awareness measurement function of other entities is set to be zero. Making use of prior knowledge is a simple and straightforward approach, but the problem is that our prior knowledge is not always reliable and it may not fit the task so well. A better way is to evaluate the relevance in a data-dependent way. For example, we can evaluate the task awareness of a relationship based on its correlation with the output.

How to design a good task awareness measurement function inevitably varies from task to task. It also depends on how the knowledge graph is used in the task. Hence, it is infeasible to find a single optimal measurement function that works well for all kinds of different tasks. Nevertheless, we can still discover some hints by analyzing the relatedness between an entity/relation/attribute and the input/output of the task. We can also find some typical methods for the most popularly studied text-related applications. For example, the task awareness of an entity in the text classification problem can be measured by classic feature selection method such as information gain.

5.3 CONSTRUCT A TASK-AWARE KNOWLEDGE GRAPH

Once the task awareness measurement function is designed, we can build a task-aware knowledge graph based on it. We need an initial knowledge graph to start with. The initial knowledge graph can be an existing general knowledge graph or a small seed graph that only contains the central entities. The initial knowledge graph may contain redundancy or noise, and its coverage may not be large enough either; the construction of task-aware knowledge graph mainly focuses on solving the two problems.

Trimming: Once we have the initial knowledge graph, the next step is to remove redundancy or noise from it based on the task awareness measurement function. The redundancy or noise can come from irrelevant or less useful entities/relations/attributes. The task awareness measurement is designed to extract the best-fit sub-graph from the initial knowledge graph. The knowledge graph is trimmed based on the awareness measurement to remove entities/relations/attributes that are not needed in the task so that the knowledge provided by the knowledge graph is less noisy.

Expansion: Expansion of the task-aware knowledge graph is made when the coverage is not large enough. Once the task awareness measurement is defined, we can have a rough sense of what kind of knowledge is useful for the task, e.g., what types of entities and relations are

most useful, what kind of attributes of entities is valuable, etc. Based on such information, we can expand the knowledge graph by extracting new entities/relations/attributes from the background text data. The well-studied entity extraction, relation extraction and other related techniques can be also applied in this step. The expansion can be made for all entities, relations and attributes, or only focuses on a small group of them. The rule of how to make expansion should be designed based on the nature of the task.

The trimming and expansion are carried out in turn until the task-aware knowledge graph stops changing, or when its coverage is sufficient meanwhile the noise is ignorable. The framework is summarized in Algorithm 5.1.

The framework provides general principles for how to construct a task-aware knowledge graph. How to design the task awareness measurement function f, the trimming method (function *trim*) and expansion method (function *expand*) is task-dependent. To make a further explanation, we illustrate an instantiation of the construction of a task-aware knowledge in Chapter 6.

Algorithm 5.1: Framework for construction of task-aware knowledge graph

Input: Task t, initial knowledge graph $G_0 = \{E_0, R_0\}$, task awareness measurement fuction f

Output: Task-aware knowledge graph $G = \{E, R, t, \Sigma\}$

$E \leftarrow E_0$;
$R \leftarrow R_0$;
$\Sigma \leftarrow \{\langle e, f(e,t)\rangle | e \in E_0\} \cup \{\langle r, f(r,t)\rangle | r \in R_0\} \cup \{\langle a, f(a,t)\rangle | a \in e \wedge e \in E_0\} \cup \{\langle a, f(a,t)\rangle | a \in r.A \wedge r \in R_0\}$;
$G \leftarrow \{E, R, t, \Sigma\}$;
repeat
  $\{E, R\} \leftarrow expand(\{E, R\}, t)$;
  $\Sigma \leftarrow \{\langle e, f(e,t)\rangle | e \in E\} \cup \{\langle r, f(r,t)\rangle | r \in R\} \cup \{\langle a, f(a,t)\rangle | a \in e \wedge e \in E\} \cup \{\langle a, f(a,t)\rangle | a \in r.A \wedge r \in R\}$;
  $\{E, R\} \leftarrow trim(\{E, R\}, \Sigma)$;
  $G \leftarrow \{E, R, t, \Sigma\}$;
until *G does not change; or the coverage is sufficient and the noise is ignorable*;

5.4 APPLY A TASK-AWARE KNOWLEDGE GRAPH

Knowledge graph is widely used in a variety of applications. In this thesis, we mainly focus on its application in text-based prediction. As shown in the study in Chapter 3, knowledge graph can be leveraged to assist featurization of text data. Compared to surface features such as n-grams, knowledge graph-based features can better capture the correlation between

the text and the response variables and thus are more effective in bridging the semantic gap, epspecially for the entity-centric prediction tasks.

Though semantic gap also exists in entity-entric text applications, the natural connection between the response variables and the central entities makes it easier to capture the correlation between textual information and the response variable with assistance of knowledge graph. In order to bridge the gap, we propose to leverage knowledge graph as an auxiliary to represent documents. Background knowledge provided by knowledge graph helps to distinguish informative textual signal from the noise, thereby better exploiting the predictive power of the textual information.

In order to leverage task-aware knowledge graph to benefit real-world text-based prediction applications, we propose to construct a knowledge graph-based featurizer to represent documents. The knowledge graph-based features can be constructed based on either entities or relations, or both. We aim at finding features that better captures the correlation between text and response variables, especially when the latter cannot be directly derivable from the former. Generally, any method that applies a generic knowledge graph in a specific task is applicable to task-aware knowledge graph because task-aware knowledge graph is essentially a knowledge graph with task awareness as a special type of weighting that indicates whether an entity or a relation should be utilized in a task.

Even though the constructed task-aware knowledge graph already helps to filter noisy information in the knowledge graph from the perspective of its applications in a particular task, the generated features can sometimes still be redundant or noisy, which may further result in overfitting problems. Therefore, a more sophisticated method for feature selection is needed in this case, which will be discussed in Chapter 7.

CHAPTER 6: CONSTRUCT A TASK-AWARE KNOWLEDGE GRAPH FOR ENTITY-CENTRIC CLASSIFICATION

In this chapter, we present an instantiation of the framework introduced in Chapter 5 on a specific family of text-based prediction applications – entity-centric classification.

As discussed before, it is hard to propose a uniform algorithm to construct and apply task-aware knowledge graph that is optimal for an arbitrary task. For example, how to define task awareness measurement function may vary from task to task. However, in the case of entity-centric prediction tasks, we can make the assumption that an entity, relation or attribute is relevant to the task only if it is related to the central entities. Since the response variables are closely related to the central entities, an entity/relation/attribute that is irrelevant to the central entities is not likely to be relevant to the response variables, thus not very helpful for the task.

We can make use of the relatedness of entity/relation/attribute to the central entities to measure the task awareness. Meanwhile, we can also make some statistical analysis such as the oc-occurrence with the central entities in the text to evaluate how an entity/relation/attribute is relevant to the task.

Based on the task awareness measurement, we can trim the knowledge graph to filter noise. For the expansion of the knowledge graph, we can find entities, relations and attributes that are related to the central entities to enlarge the coverage. More detailed example of how to design the task awareness function and how to trim and expand the knowledge graph will be introduced in the rest of this chapter.

6.1 DEFINE TASK AWARENESS MEASUREMENT FOR ENTITY-CENTRIC CLASSIFICATION

Task-aware knowledge graph can be initialized by an existing knowledge graph. The initial knowledge graph can be much larger than what we need, and only a small sub-graph of it which contains the central entities and their neighbors is relevant to the task. If we consider the knowledge graph (G) as an undirected graph where nodes are entities and edges are relations, we can compute the distance between two entities (e_i and e_j) by the shortest path: $dis(e_i, e_j, G) = min_{p \in path(e_i, e_j, G)}\{len(p)\}$ where $path(e_i, e_j, G)$ is the set of all valid paths between e_i and e_j in G (all the formal definition can be found in Section 3.2). Then a sub-graph G_{sub} that only contains the central entities and their close neighbors can be defined as:

$$G_{sub} = (\{e|\exists e', \min_{e' \in t.E_c} \{dis(e, e', G)\} \leq \theta_d\} \cup t.E, \tag{6.1}$$
$$\{r|\exists e_i \in E \wedge e_j \in t.E \wedge r \in p \wedge p \in path(e_i, e_j, G)$$
$$\wedge len(p) \leq \theta_d \wedge len(p) = dis(e_i, e_j, G)\})$$

where $t.E_c$ denotes the central entities of the task.

The sub-graph is composed of entities whose distance to at least one of the central entities is no larger than θ_d, and their shortest paths to the central entities. If an entity or relation is not included in the sub-graph, we assume that it is not very relevant to the task and set its task awareness to be zero. Taking the knowledge graph shown in Figure 3.1 as an example, the central entity is e_1 which represents the movie "La La Land". If we set θ_d as 1, then its directors, actresses and producers which are directly connected to it will be kept in the subgraph. However, the spouse of the director (e_6) will be removed because its distance to the nearest central entity is 2.

The subgraph constituted by neighbor entities of central entities and the corresponding relations that connect them provides background information about the central entities, which may be potentially valuable for the task. For example, if the task is to classify movie reviews based on the referred movies, we may have a group of relationships which directly connect a central entity (movie) to its neighbor, such as *directedBy*, *isLocatedIn*. They can tell us about features of the central entities from different aspects. Though many of them are supposed to be closely related to central entities, some of them may not be quite useful for the task. For example, *isLocatedIn* may not be as useful as some other relations such as *directedBy* for the task intuitively. In this thesis, we measure the task awareness of relations to the task in a more quantitative way.

We assume the task awareness of relations with the same type is equal. For each relation type $\mathcal{R}$, we first gather all the entities that are connected to a central entity by one of the relations of this type:

$$E_{\mathcal{R}} = \{e|e \in G_{sub} \wedge (\exists e' \in t.E \wedge r \in R \wedge \mathcal{V}(r.A_1) = \mathcal{R} \quad s.t. \quad r \in path(e, e', G))\}$$

Here we assume that relation in the knowledge graph always has type information as one of the attributes, represented as $r.A_1$. $\mathcal{V}(r.A_1)$ is the function to obtain the type of relation r. We define that a relation type $\mathcal{R}$ is mentioned in the text if and only if any of the entities in $E_{\mathcal{R}}$ occurs. If a relation type is never mentioned in the data set, the task awareness would

be set to zero.

If we can infer the distribution of labels by the mention of a relation type easily, the relation type is likely to be useful and should thus have high task awareness. Given a relation type $\mathcal{R}$, we can use the posterior probability of the class label given $\mathcal{R}$ is mentioned in the document (denoted as $p(c|\mathcal{R})$) to approximate the original distribution of labels in the data set and evaluate the approximation by KL-divergence.

We first compute the probability that the class label of a given document is c on condition that a relation type $\mathcal{R}$ is mentioned in the document:

$$p(c|\mathcal{R}) = \sum_{e \epsilon E_{\mathcal{R}}} p(c|e)p(e|\mathcal{R}) \tag{6.2}$$

where $p(e|R)$ is the probability that entity e occurs in the text given $\mathcal{R}$ is mentioned. $p(c|e)$ is the probability that the label is c given that e occurs in the text. $p(c|e)$ and $p(e|T)$ can be calculated as:

$$p(c|e) = \frac{count(c,e)}{count(e)}$$
$$p(e|\mathcal{R}) = \frac{count(e)}{\sum_{e' \epsilon E_{\mathcal{R}}} count(e')}$$

where $count(c,e)$ is how many times e occurs in a text document whose label is c, and $count(e)$ is the number of times e occured in the text data set.

Then we use the above posterior probability to approximate the original distribution of labels in the data set. If the approximation is close, the corresponding relation type is likely to be useful in the label prediction task. We can evaluate the approximation by KL-divergence:

$$D(L||\mathcal{R}) = \sum_{c \epsilon t.L} p(c) log \frac{p(c)}{p(c|\mathcal{R})} \tag{6.3}$$

where $p(c)$ is the prior probability of label c in the text data set, and $t.L$ is the set of all possible labels in the output of task t.

Equation 6.3 requires $p(c|\mathcal{R})$ to be non-zero whenever $p(c)$ is not zero. To avoid zero probability problem, we use Dirichlet smoothing to calculate $p(c|\mathcal{R})$:

$$p(c|\mathcal{R}) = \frac{\sum_{e \epsilon E_{\mathcal{R}}} count(c,e) + \mu p(c)}{\sum_{e \epsilon E_{\mathcal{R}}} count(e) + \mu} \tag{6.4}$$

where μ is a smoothing parameter to be empirically set. Note that μ should always be

non-negative ($\mu \geq 0$).

The task aware measurement of relation type $\mathcal{R}$ is finally defined as:

$$f(\mathcal{R}, t) = \begin{cases} 1/D(L||\mathcal{R}), & \text{if } |E_{\mathcal{R}}| > 0 \\ 0, & \text{else.} \end{cases} \tag{6.5}$$

The task awareness of a relation r is determined by two facts: (1) whether it is in the shortest path of an central entity to its close neighbor; (2) the task awareness of its type.

$$f(r, t) = \begin{cases} f(\mathcal{R}, t), & \text{if } r \in G_{sub} \\ 0, & \text{else.} \end{cases} \tag{6.6}$$

Entity in the knowledge graph can be considered as an ensemble of attributes. Different from the relations, attributes of entities can be mentioned directly in the text. If the mention of an attribute can help to discriminate the class labels, its task awareness should be high. We evaluate the task awareness of attributes by Gini index:

$$f(a, t) = \begin{cases} 1 - Gini(a) & \text{if } a \text{ occurs in } t.D \\ 0, & else. \end{cases} \tag{6.7}$$

where $t.D$ is the input text data.

Task awareness of an attribute is high when its distribution over all the labels is unbalanced, which means that it is a strong indicator for only a few labels. Gini index is effective especially when the number of labels is large (which may usually be the case for many entity-centric classification tasks). Other kinds of measurement such as information gain and χ-square can also be used. The task awareness of an entity is measured based on its attributes:

$$f(e, t) = \begin{cases} max_{a \in e}\{f(a, t)\}, & \text{if } e \in G_{sub} \\ 0, & else. \end{cases} \tag{6.8}$$

If an entity is a close neighbor of any of the central entities, its task awareness is the maximal value of the task awareness of all its attributes. Otherwise, its task awareness is set to be zero.

6.2 EXPAND KNOWLEDGE GRAPH BY FINDING NEW ENTITIES AND RELATIONS

A main deficiency of using pre-constructed knowledge graph in a static way is that the coverage is insufficient in many cases, which eventually hurts the performance. Informative signals from the text may be ignored due to the absence of the required background knowledge. To solve this problem, we propose to expand the knowledge graph from the background text data in the task. Some of the entities or relations that are helpful for the task could be extracted from the text corpus, but are actually missing in the knowledge graph. If we can discover these unexplored entities and relations from the text corpus, the coverage will be enlarged, thus the power of knowledge graph will be better exploited.

We first extract entities from the corpus by NLP tools to enrich the entity set. The next step is to identify which of the newly found entities are related to the existing entities in the knowledge graph, especially the central entities. To achieve this goal, we use a joint embedding model to represent all entities and relations in the same vector space so that we can calculate the probability that a new entity is connected to an existing entity by a certain type of relation.

The joint embedding model is constituted of two models. One is a knowledge graph embedding model which learns a low-dimensional vector for the entities and relations in the same vector space. The knowledge graph embedding model we use is similar to TransE [49]. The basic assumption that the tail entity vector should be close to the sum of the head entity vector and the relation vector. Thus, the head/tail entity can be estimated given the tail/head entity and the relation, while the relation can be also inferred given the head and the tail entities. The goal of the knowledge graph embedding is to minimize the estimation error. More specifically, the knowledge graph embedding is learned by minimizing the following objective function:

$$\begin{aligned}\mathcal{L}_{\mathcal{G}} = \sum_{\langle e_i,r,e_j\rangle\in R} \{ & \sum_{e'\in\Omega(e_i,r)} [\gamma + d(u_{e_i} + u_r, u_{e_j}) - d(u_{e_i} + u_r, u_{e'})]_+ \\ & + \sum_{r'\in\Omega(e_i,e_j)} [\gamma + d(u_{e_i} + u_r, u_{e_j}) - d(u_{e_i} + u_{r'}, u_{e_j})]_+ \\ & + \sum_{e'\in\Omega(r,e_j)} [\gamma + d(u_{e_i} + u_r, u_{e_j}) - d(u_{e'} + u_r, u_{e_j})]_+ \} \end{aligned} \tag{6.9}$$

where u_{e_i}, u_r and u_{e_j} are the vectors for the head entity, the relation and the tail entity respectively. $d(u_i, u_j) = \frac{1}{2}\|u_i - u_j\|_2^2$ is the distance between two vectors u_i and u_j. $\Omega(e_i, r)$ are the negative examples for entity e_i and relation r where an negative example e' does not hold relation r with entity e_i. Similarly, $\Omega(r, e_j)$ are the negative sampling for retrieving the

head entities. $\Omega(e_i, e_j)$ is the negative examples of the relations.

Knowledge graph embedding models have been shown to be promising in knowledge graph completion and have attracted much attention recently [50]. However, the scope is still limited to the entities that are already included in the knowledge graph. We believe that learning knowledge from the task can help to enhance the knowledge graph, and the expansion of the knowledge graph can also benefit from leveraging the text data. For example, reviews like "City of Stars is my favorite song in La La Land" may indicate that the entity "La La Land" is connected to "City of Stars" by "hasSong" relation. In order to extend the coverage to entities that are out of the knowledge graph, we propose another model for text embedding to learn the vector representation for entity in the text corpus. Inspired by Skip-gram, we also predict the context words/entities based on the target word/entity. When we see a word occurs in the text, the probability of seeing its context words is estimated in the same way as in Skip-gram.

Once we find an entity mentioned in a training document, we assume that the entity is implicitly related to the corresponding central entiti(es) which is assigned as the label(s) of the document. For example, if we see "City of Stars" occurs in a review about the movie "La La Land", then these two entities may be connected by a certain type of relationship. This assumption is different from traditional joint embedding techniques (e.g., [31]) which requires the related entity to co-occur together in the same context. For a random text collected from an arbitrary source, we can only rely on the explicit oc-occurrence to extract relations. However, in the circumstance of entity-centric classification, the constraint can be more flexible. Based on this assumption, we can infer the context of the mentioned entity by its relation to the central entiti(es). We assume that the entities that have the same type of relation with a central entity are likely to see similar words in their context. For example, when we see a song of a movie is mentioned in the text, we may expect to find contextual words such as "song" and "composer", while words like "performance" are likely to be found in the context of the actor/actress. Formally, the text embedding is learned by:

$$\mathcal{L}_{\mathcal{T}} = \sum_{(w,w_c)\in D} \log P(w_c|w) + \sum_{(e,w_c)\in D} \log P(w_c|e_c, r)P(r|e_c, e) \tag{6.10}$$

$$P(w_c|w) = \frac{v_{w_c} \cdot u_w}{\sum_{w'_c} v_{w'_c} \cdot u_w} \tag{6.11}$$

$$P(w_c|e_c, r) = \frac{v_{w_c} \cdot (u_{e_c} + u_r)}{\sum_{w'_c} v_{w'_c} \cdot (u_{e_c} + u_r)} \tag{6.12}$$

$$P(r|e_c, e) = \frac{exp(q(e_c, r, e))}{\sum_{r'} exp(q(e_c, r', e))} \tag{6.13}$$

where v_w and u_w represent the output and input vector for word w respectively. $q(e_c, r, e) = \alpha - \frac{1}{2}\|u_{e_c} + u_r - u_e\|_2^2$ is the normalized distance between e_c and e given that the relation between them is r. Negative sampling is used to approximate the softmax function in Equation 6.11 and 6.12.

When we obtain the embedding vectors, we can discover the related entities to a central entity e_c by a certain relationship r based on the following probability:

$$P(e|e_c, r) = \frac{exp(q(e_c, r, e))}{\sum_{e'} exp(q(e_c, r, e'))} \tag{6.14}$$

The primary of the expansion is to discover unknown entities and relations that are related to the central entities so that we can enlarge the coverage of useful knowledge for the task. Based on the background text corpus, we are able to find new entities that are not covered by the knowledge graph. Among all of these entities, some of them are actually closely related to the central entities, while the others may not very relevant to the central entities and thus is less likely to be potentially useful for the task. For each central entity e_c, we only collect the newly found entities that are most likely to be related to it and add them as its neighbors. In order to find such closely related entities, we measure the probability of a new entity being connected to the central entity e_c by relation r by Equation 6.14.

Besides discovering new entities and relations, we also expand the knowledge graph by finding aliases for the entity names.

We mainly use the task-aware knowledge graph for knowledge graph-assistant featurization in this task. More specifically, we can extract all the attributes (e.g., entity names) from the text and represent the text as a vector where each dimension is an attribute or a combination of an attribute and its context words. The quality and coverage of the attributes, especially entity names, will have a great impact on the performance of the task. If the knowledge graph can cover most of the aliases mentioned in the data set, it will help to classify the text more accurately. Thus, we can also use the background text corpus to find aliases of

entity names for further expansion of the knowledge graph. Because it is carried out in an automatic way, noise or errors might be brought in. To control the quality, task awareness measurement is used to filter out the newly found aliases which contribute little to the classification task.

Most of the entity names provided by an existing general knowledge graph are full names. However, the entity is sometimes mentioned as an abbreviation. For each entity name, we extract all its sub-phrases composed of adjacent words in it as alias candidates. If an original name is $\langle w_1, w_2, ..., w_n \rangle$, a valid sub-phrase is like $\langle w_i, w_{i+1}, ..., w_{i+m} \rangle$ where $1 \leq i + 1$ and $i + m \leq n$. When a candidate alias is extracted from the original name, we match it in the text data set. If an alias never occurs, it indicates that the alias we find is not correct or is not useful for the task.

6.3 REFINE TRIMMING AND EXPANSION FUNCTIONS

The trimming function we designed for the task is simple. It removes entities, relations, and attributes whose task awareness is lower than a threshold. After a primary trimming is made to remove irrelevant entities and relations from the initial knowledge graph, the next step is to expand it.

The expansion is made by finding new entities, relations and aliases for entity names based on the background text corpus. Given a new entity, if its probability of being connected to any of the central entities by a certain relation is above a cut-off threshld (θ_p), it is added to the knowledge graph together with the corresponding relation. The trimming and expansion function is plugged in the framework proposed in Chapter 5.

Finally, we get a set of entities and relations with their task awareness. They together form the task-aware knowledge graph. The full process is summarized in Algorithm 6.1. θ_r. θ_e and θ_a are the thresholds used to filter relations, entities and attributes respectively. Multiple task-aware knowledge graphs can be generated with different thresholds. Lower threshold enables larger coverage, but also makes the feature noisier if we use the task-aware knowledge graph for text representation. Higher threshold finds better features but may hurt the coverage. In practice, we can tune the threshold parameters based on the data set to make a better trade-off between the coverage and the quality.

The above algorithm is only one example of how the general framework can be instantiated for the entity-centric classification tasks. Note that the framework can be instantiated in many other ways as well.

Algorithm 6.1: Construct task-aware knowledge graph for entity-centric classification tasks

Input: Task t, initial knowledge graph $G_0 = \{E_0, R_0\}$, task awareness measurement fuction f

Output: Task-aware knowledge graph $G = \{E, R, t, \Sigma\}$

$E \leftarrow E_0$;
$R \leftarrow R_0$;
$\Sigma \leftarrow \{\langle e, f(e,t)\rangle | e \in E_0\} \cup \{\langle r, f(r,t)\rangle | r \in R_0\} \cup \{\langle a, f(a,t)\rangle | a \in e \wedge e \in E_0\} \cup \{\langle a, f(a,t)\rangle | a \in r \wedge r \in R_0\}$;
$\{E, R\} \leftarrow trim(\{E, R\}, \Sigma)$;
$G \leftarrow \{E, R, t, \Sigma\}$;
repeat
  $\{E, R\} \leftarrow expand(\{E, R\}, t)$;
  $\Sigma \leftarrow \{f(e,t) | e \in E\} \cup \{f(r,t) | r \in R\} \cup \{f(a,t) | a \in e \wedge e \in E\} \cup \{f(a,t) | a \in r \wedge r \in R\}$;
  $\{E, R\} \leftarrow trim(\{E, R\}, \Sigma)$;
  $G \leftarrow \{E, R, t, \Sigma\}$;
until *G does not change; or the coverage of G is sufficient and the noise in G is ignorable*;
Function $trim(E, R)$
  $e \leftarrow \{a | a \in e \wedge \langle a, f(a,t)\rangle \in \Sigma \wedge f(a,t) > \theta_r\}$;
  $E \leftarrow \{e | e \in E \wedge \langle e, f(e,t)\rangle \in \Sigma \wedge f(e,t) > \theta_e\}$;
  $R \leftarrow \{e | e \in R \wedge \langle r, f(r,t)\rangle \in \Sigma \wedge f(r,t) > \theta_a\}$;
Function $expand(E, R)$
  foreach $e \in E_0$ **do**
    if $\exists e_c \in t.E_c, r = \langle e_c, r.A, e\rangle \quad s.t. \quad r.A_1 \in \{r'.A_1 | r' \in R_0\} \wedge P(e|e_c, r) > \theta_p$ **then**
      $E \leftarrow E \cup \{e\}$;
      $R \leftarrow R \cup \{r\}$;
    end
  end
  $e \leftarrow \{a | a \in e\} \cup \{\langle$ *"name"*$, alias\rangle | alias \in sub\_phrases(\mathcal{V}(a)) \wedge a \in e \wedge \mathcal{K}(a) =$ *"name"*$\}$;

6.4 EXPERIMENT

In this section, we apply the method introduced in this chapter in a movie review classification task. The problem is to automatically annotate movie reviews with the movies being reviewed. The movie label belongs to a pre-given set, which are the central entities of the task. The experiment setup is the same as the movie classification experiment discussed in Chapter 3

6.4.1 Performance of Task-aware Knowledge Graph

The primary goal of constructing a task-aware knowledge graph is to benefit the real-world applications. A natural way to evaluate a task-aware knowledge graph is to estimate its performance in the task. In our experiments, we use four metrics to evaluate the performance: accuracy, precision, recall and F-1 measure.

We use two generic knowledge graphs in the movie review classification task. One is built from Wikipedia Infobox and meta data (denoted as "Wiki"), and the other is Yago. According to our study in Chapter 3, entity-based features (EF) and context-aware entity-based features (CaEF) work better in the classification task. Hence, we employ these two featurization method to construct knowledge graph-based features.

As shown in Table 6.1, the generic knowledge graphs indeed help to improve the performance. Lexical features such as n-grams are very noisy so the precision is lower than the generic knowledge graphs, even if L-1 regularization is employed. The generic knowledge graphs also outperform syntactic features such as the POS tags ("POS") and dependency triples ("DEP"). Among all the baseline methods, the extracted named entity works best, even better than the generic knowledge graphs sometimes, due to the larger coverage compared to the pre-constructed knowledge graphs.

The performance of the knowledge graphs is far from optimal since the coverage of the generic knowledge graphs is not large enough for this task. Actually the performance gets significantly improved because of the enlargement of the coverage when the knowledge graphs are expanded by paradigmatic or syntagmatic relations (e.g., "Wiki + $Para_{sg}$ " which denotes the knowledge graph built from Wiki Infobox and meta data by paradigmatic relations mined by Skip-gram model). When we combine the entity-based features generated based on the original knowledge graphs with the extracted named entities ("Wiki + NER" and "Yago + NER"), it works better than using them separately, but are not as effective as directly expanding the knowledge graph by adding more related entities because the recognized named entities are filtered by the paradigmatic or syntagmatic relations so that the features are less noisy. Again, we find that the knowledge graph-based features are more effective with the context information, for both the original knowledge graphs and the expanded knowledge graphs.

Next, since we have now confirmed that knowledge graph is helpful in this task and expansion of the knowledge graph even makes the prediction more accurate, we turn to ask the central research question whether a task-aware knowledge graph can perform even better. To construct the task-aware knowledge graph, we set θ_e and θ_a to be 0.05, and θ_r to be 0.135. θ_p is set to be 0.1 for the knowledge graph built from Wiki and 0.05 for Yago.

Table 6.1: Baseline method v.s. knowledge graph

feature	accuracy	precision	recall	F-1 measure
unigram	0.374	0.371	0.362	0.366
uni- and bigram	0.378	0.379	0.365	0.372
uni-, bi- and trigram	0.380	0.376	0.367	0.371
POS	0.270	0.249	0.243	0.246
DEP	0.171	0.158	0.152	0.155
NER	0.327	0.448	0.323	0.376
W2V	0.193	0.181	0.177	0.179
Wiki (EF) + NER	0.438	0.545	0.429	0.480
Yago (EF) + NER	0.374	0.467	0.358	0.405
Wiki (EF)	0.339	0.419	0.332	0.370
Wiki + $Para_{sg}$ (EF)	0.443	0.555	0.434	0.487
TAKG_Wiki (EF)	**0.455**	**0.571**	**0.450**	**0.503**
Yago (EF)	0.303	0.439	0.296	0.353
Yago + Syn_{rw} (EF)	0.443	0.547	0.433	0.484
TAKG_Yago (EF)	0.366	0.507	0.365	0.425
Wiki (CaEF)	0.532	0.511	0.513	0.512
Wiki + $Para_{sg}$ (CaEF)	0.555	0.549	0.543	0.546
TAKG_Wiki (CaEF)	**0.572***	**0.586***	**0.563***	**0.574***
Yago (CaEF)	0.451	0.437	0.432	0.435
Yago + $Para_{sg}$ (CaEF)	0.493	0.495	0.480	0.487
TAKG_Yago (CaEF)	0.455	0.473	0.447	0.460

Two-tailed t-test is done for paired data. In each pair, one is GKG, and the other one is any of the other methods. * indicates $p\text{-}value < 0.01$ for all tests.

μ is a smoothing parameter that we simply set to a small value (0.001) without tuning. Parameter θ_d in Equation 6.1 is set to be 1. The performance is shown in Table 6.2.

TAKG is the task-aware knowledge graph we construct. TAKG_Wiki is the task-aware knowledge graph constructed based on the generic knowledge graph built from Wiki Infobox, while TAKG_Yago uses Yago as the initial knowledge graph. We can see that the task aware knowledge graphs work better than both of the generic knowledge graphs because the task-aware knowledge graphs are augmented by learning new knowledge from the application. By enlarging the coverage and reducing the noise, the entity features from task-aware knowledge graphs are more informative.

The expansion of the knowledge graph in the construction of task-aware knowledge graph is more customized compared to the expansion based on paradigmatic and syntagmatic relations. When the generic knowledge graph provides enough example of related entities to

learn, task-aware knowledge graphs are more effective in generating good features and thus achieve better performance. In fact, the knowledge graph built from Wiki Infobox and the meta data has much better coverage than Yago. The former covers all the central entities while over one third of the central entities are missing the latter. As a result, TAKG_Wiki works better than the expanded knowledge graph based on paradigmatic or syntagmatic relations. Paradigmatic relation mined by Skip-gram model is the most effective version in the paradigmatic/syntagmatic relation-based expansion for the knowledge graph built from Wiki Infobox, and its performance ("Wiki + $Para_{sg}$") is worse than the task-aware knowledge graph ("TAKG_Wiki"). For Yago, however, expanding the knowledge graph based on paradigmatic or syntagmatic relation helps better in the task compared to constructing a task-aware knowledge graph, because the original knowledge graph only covers limited number of central entities and consequently the embedding algorithm proposed in Section 6.2 is not as effective as expected to find new entities and relations that are relevant to the task.

According to our study, the related entities are the most useful predictors of the mentioned movies in the reviews. In fact, the entity-based features are not only effective but also "sufficient" for the task, and adding other features on top of it may even bring in noise which leads to overfitting problem and finally hurts the performance. For example, if we add the unigrams on top on the entity-based features generated based on the task-aware knowledge graph built from Wiki Infobox, the F-1 measure drops to, and the decrease is mainly due to the overfitting problem. We test "TAKG_Wiki (EF)" with unigrams on a subset of *training* data, and it indeed has much high value on all three metrics (e.g., 1.0 on F-1 measure). The fact that it does not work well on the test set clearly shows that the unigrams indeed have caused overfitting when the unigram features are added to task-aware knowledge graph-based features. These results also suggest that the more generalizable features generated by the task-aware knowledge graph tend to "lose" to more specific features like unigrams in the training process due to the high potential of overfitting the training examples by those unigram features. In general, all the results show that the task-aware knowledge graphs and the expanded knowledge graphs have a sufficient coverage for the task while containing little noise compared to the original knowledge graphs and baseline methods.

To see how the change of task-aware knowledge graph during its construction makes influence on its application, we investigate each step of trimming and expansion. We apply the task-aware knowledge graph built in each step in the task. Entity-based features are used. Logistic regression without $L1$ regularization is employed to avoid automatically filtering of noisy features so that we can have a better estimation of how noisy the task-aware knowledge graph can be.

We start from the initial knowledge graph that constructed from Wiki Infobox and the

Table 6.2: Analysis of impact of each step in the construction of task-aware knowledge graph

feature	accuracy (gain)	precision (gain)	recall (gain)	F-1 measure (gain)
Wiki (EF)	0.318	0.387	0.305	0.341
First Trimming	0.289 (-0.029*)	0.446 (0.059*)	0.284 (-0.021*)	0.347 (0.006*)
First Expansion	0.432 (**0.143***)	0.543 (**0.097***)	0.421 (**0.137***)	0.474 (**0.127***)
Second Trimming	0.450 (0.018*)	0.559 (0.016*)	0.442 (0.021*)	0.494 (0.020*)

One-tailed t-test is done for the gain of the performance, and * indicates $p\text{-}value < 0.01$ for all tests.

meta data in Table 6.2. The first trimming removes entities that are not directly connected with central entities and their relations. Relations and entities whose task awareness is less than the thresholds are filtered as well (Line 4 in Algorithm 6.1). After the first trimming, only 4 out of all the 17 relationships are kept in the knowledge, namely, *starredBy*, *directedBy*, *writtenBy* and *producedBy*. It is in line with our common sense that it is easy to infer which movie a review is about when its directors, actors or authors are mentioned in the text. We can see that because we narrow down the scope of background knowledge to the best-fit types, the precision is increased after the first trimming. However, the coverage is lowered by filtering out so many entities and relations, which consequently hurts the recall.

The first expansion is done by adding new entities, relations and aliases of entity names (Line 7 in Algorithm 6.1), leading to the largest increase in all metrics because the coverage of knowledge is enlarged significantly. The second trimming reduces noise brought in by the expansion (Line 9 in Algorithm 6.1) Because the noise is filtered by this step, the performance is further improved.

Similar conclusion can be drawn from comparison of each step in the construction process of the task-aware knowledge graph built from Yago.

Though continuous improvement is made by polishing the task-aware knowledge graph step by step, the overall performance is still far from ideal. The evaluation results are not very high in general. Apart from the imperfection of the task-aware knowledge graph, another important reason is that the movie review classification is not an easy task in general, not only for machine but also for human beings. It is because some reviews are not able to provide enough hints for the inference of movie labels. For example, the following review is

about the movie "*The Ground Truth*":

- *"Anyone who claims to support the troops owes it to them to see the film and hear their stories."*

No identification information about the related movie is mentioned in the review, and it is almost impossible to know which movie it is about without any additional information. To have a quantitative evaluation of the difficulty of the task. We randomly sample 100 reviews from the corpus and manually label them. The best accuracy human annotators can get is 0.57, which can be considered as an approximation of the upper bound of the performance for this task. In this sense, our method gets surprisingly good performance given that the best accuracy the task-aware knowledge graph can achieve is around 0.572 ("TAKG_Wiki (CaEF)").

6.4.2 Impact of thresholding

To check the sensitivity of performances to the parameter setting, we can generate multiple versions of task-aware knowledge graph by using different thresholds and apply them in the task. We can also have an insight into how the selection of relationships and the restrictions in expansion can influence the task.

There are four thresholds, namely, θ_p, θ_r, θ_e and θ_a. θ_p controls the expansion process. With small θ_p, more newly found entities and relations will be added to the knowledge graph, which results in larger coverage but may also hurt the quality because of more noise. The task-aware knowledge graph reaches a smaller size with larger θ_p, but the expansion could be more reliable. To investigate the impact of θ_p, we construct variations of the task-aware knowledge graph by only changing θ_p. The generic knowledge graph built from Wiki is used to initialize the task-aware knowledge graph. Entity-based features are used in linear regression without L-1 regularization.

As shown in Figure 6.1, the performance is not sensitive to θ_p. When θ_p increases, the overall performance is neither improved nor hurt. A main reason is that even though high θ_p may filter many of the related entities, the discovery of aliases will make substantial expansion in the other way. However, we may not want θ_p to be too small since it causes considerable noise. The performance of the task-aware knowledge graph based on Yago is stable over different θ_p's as well.

θ_r is used to filter relations. The 17 relation types in the initial knowledge graph are ranked by their task awareness. We can have different combinations of top relationships by changing θ_r. To show the impact of θ_r, we construct multiple versions of TAKG, each

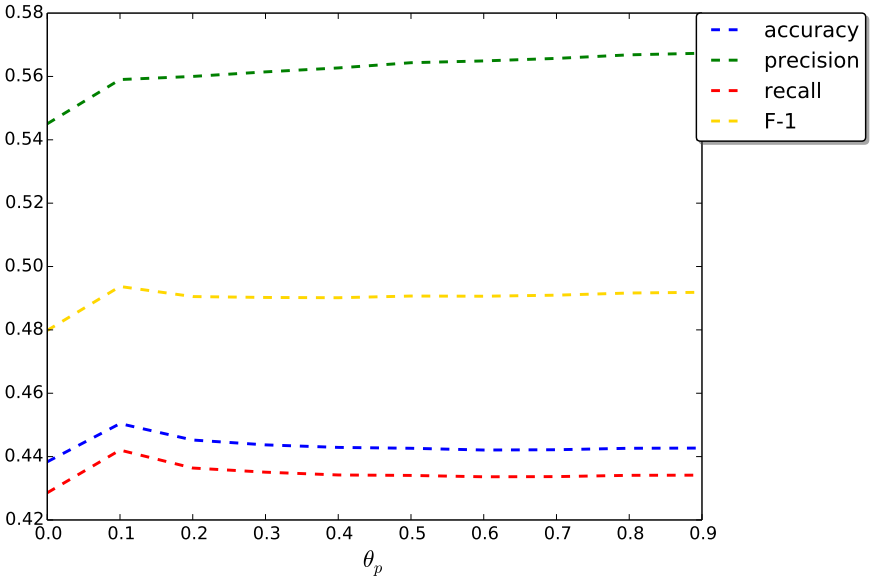


Figure 6.1: Impact of θ_p.

Table 6.3: Analysis of impact of relation types

K	Top K relation types	accuracy	precision	recall	F-1 measure
0	Wiki (EF)	0.318	0.387	0.305	0.341
1	*starredBy*	0.388	0.491	0.381	0.429
2	*starredBy*, *directedBy*	0.421	0.521	0.413	0.461
3	*starredBy*, *directedBy*, *writtenBy*	0.426	0.526	0.417	0.465
4	*starredBy*, *directedBy*, *writtenBy* *producedBy*	0.432	**0.543***	0.421	**0.474**
5	*starredBy*, *directedBy*, *writtenBy*, *producedBy*, *distributedBy*	**0.439***	0.532	**0.429***	**0.474**

Two-tailed t-test is done for paired data. In each pair, one is the version gets the best performance, and the other one is any of the other versions. * indicates $p\text{-}value < 0.01$ for all test.

containing different types of relations. The performance can be found in Table 6.3. θ_a and θ_e are both fixed to be 0 to keep all the relations of the selected type. Logistic regression without $L1$ regularization is used again.

If we start with the generic knowledge graph built from Wiki Infobox and meta data, the top four relationship are *starredBy*, *DirectedBy*, *writtenBy* and *producedBy*. Relationship *starredBy* gets the highest task awareness. If we only use it, the task-aware knowledge graph is then composed of movie entities connected with their actors. It has the smallest coverage in all the listed versions of task-aware knowledge graph, so it gets the lower recall

compared to the others. However, because it is still augmented by expansion, it outperforms the original generic knowledge graph on all metrics. When we add *directedBy*, *writtenBy* and *producedBy* relationships in the task-aware knowledge graph, the performance is further improved. It is because director and author are hot topics extensively discussed in movie reviews. As the number of relation types grows, the coverage increases but noise is also brought in. When more relations like *distributedBy* are included, the recall rises but the precision drops. The trend is similar for larger K's.

When constructing task-aware knowledge graph from Yago, the top 3 relationships are *ActedIn*, *Edited* and *Directed* respectively, which is close to the ranking in the generic knowledge graph built from Wiki. We can again find the similar trend in variations of TAKG_YAGO that with more relationships included, the features becomes noisier and thus hurts the performance.

Two parameters control the growth of entities and their attributes, i.e., θ_e and θ_a respectively. By changing θ_e and θ_a, we can have another group of task-aware knowledge graphs. We set θ_e and θ_a as the same (θ_a is dominant to θ_e), and the performance of the generated task-aware knowledge graphs is shown in Figure 6.2. Small θ_a and θ_e are favored in general. Large threshold makes the quality better controlled but results in poor coverage which hurts the performance. Small threshold enables more aggressive expansion but may also raise the risk of bringing in noise. Considerable noise is brought in and the performance gets worse when θ_a and θ_e become too low (e.g, 0.01 or 0.02). The overall performance is more sensitive to θ_e and θ_a compared to θ_r, because large θ_e and θ_a hurts the coverage more significantly than large θ_r.

Small θ_a and θ_e are favored in general. Large threshold makes the quality better controlled but results in poor coverage which hurts the performance. Small threshold enables more aggressive expansion but may also raise the risk of bringing in noise. Considerable noise information is brought in and the performance gets worse when θ_a and θ_e become too low (e.g, 0.01 or 0.02). The overall performance is more sensitive to θ_e and θ_a compared to θ_r, because large θ_e and θ_a hurts the coverage more significantly than large θ_r.

6.4.3 Summary

To summarize, the constructed task-aware knowledge graph is indeed effective in improving the performance of the task. The gain mainly comes from expansion but trimming also helps to make further improvement. In general, it is beneficial to make use of knowledge graph. However, task-aware knowledge graph outperforms general knowledge graphs because it provides customized knowledge for the task. With adequate training examples to learn the

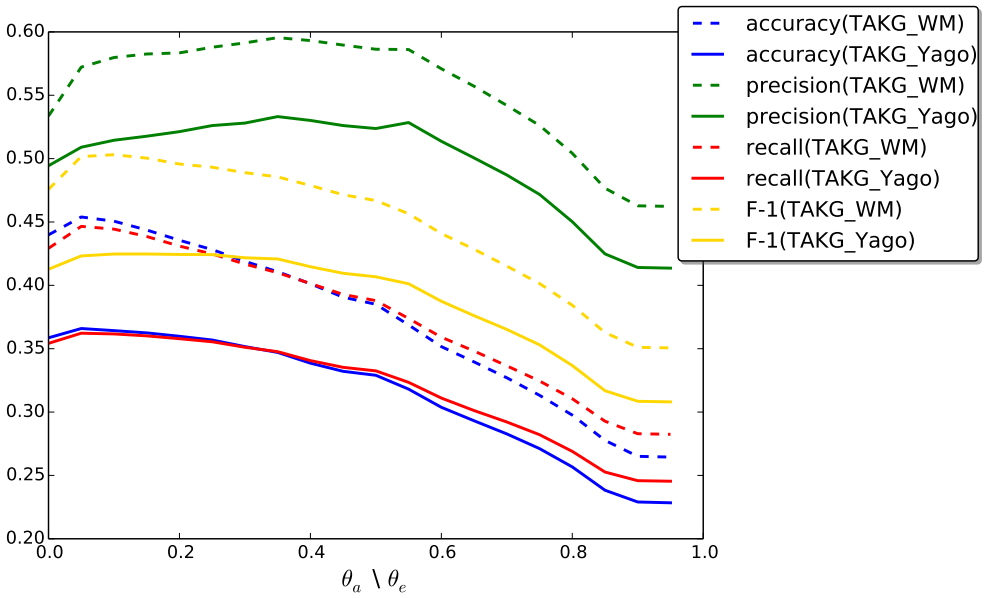


Figure 6.2: Impact of θ_e and θ_a.

embedding vectors for the entities and the relations, the task-aware knowledge graph can have a better coverage of the entities or relations relevant to the task and consequently works better than the knowledge graph expanded by more general word relations such as paradigmatic or syntagmatic relations. Generally, when the coverage of the central entities in the existing generic knowledge graph is good enough, finding new entities and relations by the jointly embedding model is more promising. While when the coverage of the central entities is poor, making use of more "vague" relations such as paradigmatic and syntagmatic relations is more beneficial.

The task awareness function helps to find the most useful types of relations. By removing the noisy features brought in by irrelevant relations, the performance can be improved. Besides, this kind of finding can also cast insight into the interpretation of what kind of information is most useful for identifying the class labels. For example, we find in our experiment that the actor/actress and directory information is important, and a review is probably relevant to the corresponding movie when they are mentioned in it. Such kind of knowledge may also be applicable in another task with the same type.

Trimming of entities and attributes is also necessary because sometimes the expansion introduces noise. In general, low threshold is favored for the trimming of attributes and entities if enlarging the coverage is more important for the task. But it should not be too low. Otherwise, the quality of the knowledge graph will be hurt.

Context-aware entity-based featurization works best when applying the task-aware knowl-

edge graph (or other types of knowledge graph such as a generic knowledge graph or an expanded version) in the entity-centric classification task which takes the central entities as class labels. In general, entity-centric classification is more challenging than thematic categorization of documents because background knowledge is required to derive class labels from the text. By constructing a task-aware knowledge graph and applying it in knowledge graph-assistant featurization, the performance can even be close to that a human annotator can reach. Though the performance is still not perfect, it is promising that the construction and application of task-aware knowledge graph can benefit real-world applications.

CHAPTER 7: APPLY KNOWLEDGE GRAPH FOR ENTITY-CENTRIC REGRESSION

In this chapter, we present the proposed approach for exploiting knowledge graph to improve entity-centric prediction [51]. The construction process of the task-aware knowledge graph is the same as that shown in Chapter 6, because we believe that identifying the related information about the central entities in a sentence or paragraph is the primary step to mine deeper semantic information. As discussed in Chapter 3, context-aware explanatory path-based features works best for the regression task, thus we employ it as the featurization method in this chapter. The experimental results in our previous study (see Chapter 3) show that the extracted context-aware explanatory path-based feawtures are still very noisy and we need further pruning to make sure that the signal information can stand out. To solve this problem, we present a two-stage filtering algorithm to improve feature representation.

7.1 FEATURIZATION

As discussed in Chapter 3, the explanatory path together with the contextual information generates better features that capture the correlation between text input and the response variables compared to other kinds of knowledge graph-based features. Entities that are related to the central entities in the same way may play a similar role in the predictive analysis, and merging them together helps to alleviate the sparsity problem in the feature space. Moreover, relation types are much more prevalent than individual entities, so they can be easily adapted to new data.

Based on the findings in the study about the knowledge graph-based feature engineering (see Chapter 3), we refine the context-aware explanatory path-based features by allowing n-grams extracted from both the left and right side. To further generalize the extracted features for varied central entities, we remove the central entity which is at the end of the explanatory path. In this way, the features is generalizable to unseen central entities as well. More specifically, the features can be defined as:

$$\begin{aligned}\mathcal{K}(s,e,l,W) = \{(\langle w'_{i+1},\ldots,w'_{i+l_1}\rangle, \{\mathcal{T}(p)|e' \in E_c \wedge p \in path(e,\\ e',G) \wedge path(e,e',G) = dis(e,e',G)\}, \langle w'_{q+1},\ldots,w'_{q+l_2}\rangle)|(\langle w'_{i+1},\\ \ldots,w'_{i+l_1}\rangle, e.n, \langle w'_{q+1},\ldots w'_{q+l_2}\rangle) \in \mathcal{E}(s,e,l,W)\}\end{aligned} \tag{7.1}$$

where l is the length of the contexual phrase which can be from either left or right side or both. W is the maximum context window size. $\mathcal{T}(p)$ is a function to get all the relation

types in path p and $\mathcal{T}(\langle r_1, r_2, \ldots r_l \rangle) = \langle r_1.T, r_2.T, \ldots, r_l.T \rangle$. Note that if the entity e is one of the central entities, then $\mathcal{T}(p) = \langle \text{“}SELF\text{”} \rangle$.

$\mathcal{E}(s, e, l, W)$ is the context-aware features extracted from sentence s given an entity e:

$$\begin{aligned}\mathcal{E}(s, e, l, W) = \{(\langle w'_i, \ldots, w'_{i+l_1} \rangle, \langle w'_j, \ldots, w'_{j+k-1} \rangle, \langle w'_q, \ldots, \\ w'_{q+l_2} \rangle) | e.n = \langle w_1, \ldots, w_k \rangle \wedge s = \langle w'_1, \ldots, w'_{|s|} \rangle \wedge w'_j = w_1 \wedge \\ \ldots \wedge w'_{j+k-1} = w_k \wedge i + l_1 < j \wedge j - i \leq W \wedge q + l_2 \leq |s| \wedge \\ j + k - 1 < q \wedge q + l_2 - (j + k - 1) \leq W \wedge l_1 + l_2 = l\} \end{aligned} \tag{7.2}$$

where $e.n$ is the name of entity e. The context n-gram does not have to be adjacent with the entity mention.

Example 7.1. Given the relation between “Rowling” and the central entity “Harry Potter and the Order of the Phoenix” is “WrittenBy”, let l and W both be 2, then the context-aware explanatory path-based features extracted from the sentence “*Goblet of Fire is the entry in which Rowling finally took off the gloves.*” are: $\mathcal{K}(s, e, l, W) = \{(\langle$ “in” “which”$\rangle$, $\langle$“WrittenBy”$\rangle$, $\langle\rangle$), ($\langle\rangle$, $\langle$“WrittenBy”$\rangle$, $\langle$ “finally”, “took” $\rangle$), ($\langle$ “in” $\rangle$, $\langle$ “WrittenBy” $\rangle$, $\langle$ “finally” $\rangle$) , ($\langle$ “in” $\rangle$, $\langle$ “WrittenBy” $\rangle$, $\langle$ “took” $\rangle$), ($\langle$ “which” $\rangle$, $\langle$ “WrittenBy” $\rangle$, $\langle$ “finally” $\rangle$), ($\langle$ “which” $\rangle$, $\langle$“WrittenBy”$\rangle$, $\langle$“took”$\rangle$)$\}$

By setting the contextual phrase length (l in Equation 3.10) to different values, we can get multiple sets of context-aware explanatory path-based features. Let L be all possible settings of contextual phrase length we want, then the context-aware explanatory path-based features in a document d are:

$$\mathcal{K}(d, E) = \cup_{s \in d, e \in E, l \in L} \mathcal{K}(s, e, l, W) \tag{7.3}$$

7.2 PRIMARY FILTERING OF FEATURES

The generated context-aware explanatory path-based features can better capture the correlation between the text and the response variable than surface lexical features. However,

the original features are quite noisy, and it is necessary to filter the features in a data-dependent way to improve the quality.

To this end, we define a t-statistic based measure to filter the features. Given a feature f, we first find all the documents that contain it and denote the set as D_f. Then align the documents with the response variable: $Y_f = \{y_i | d_i \in D_f\}$. The t-statistic based measure is defined as:

$$ts(f) = \frac{mean(Y_f) - mean(Y)}{\frac{std(Y_f)}{\sqrt{|Y_f|}}} \tag{7.4}$$

where $mean(Y_f)$ is the mean of Y_f and $std(Y_f)$ is its standard deviation. Y is the response for all the documents.

Suppose we make a hypothesis that the true parameter of Y_f equals to Y, then Equation 7.4 can be used to perform a t-test with degrees of freedom of $|Y_f|-1$, and the hypothesis is more likely to be accepted in general when the value is small. Consider Y_f as the response variables conditional on feature f. When the true distribution of Y_f equals to Y (the hypothesis is accepted), it means that the uncertainty of Y is not reduced by f and f is not so useful for prediction. Think about two features – one is a common word that occurs in every document, while the other can only be found in documents that have the largest response variable in the dataset. The distribution of Y_f for the first one is exactly the same as that of Y whereas the second one is more likely to be an effective feature because its occurrence indicates a large value of the response variable.

We can also interpret the t-statistic measure in another way. Initially, we normalize all the response variables as:

$$\hat{Y} = [\hat{y_1}, \ldots, \hat{y_n}]^T = [y_1 - mean(Y), \ldots, y_n - mean(Y)]^T$$

Then $\hat{Y_f} = \{\hat{y_i} | d_i \in D_f\}$ and $ts(f) = \frac{mean(\hat{Y_f})\sqrt{|\hat{Y_f}|}}{std(\hat{Y_f})}$.

We can see a feature will have a high t-statistic measure when its normalized response variables have high average or low variation. Low variation means that the feature is likely to be an effective indicator for a small range of response variables, while high-average features are more robust to outliers. $\sqrt{|\hat{Y_f}|}$ in the numerator penalizes low-frequency features and rewards high-frequency ones whose performance tends to be more reliable in unseen examples.

Compared to other kinds of measurements such as Chi-square, the t-statistic based measure not only works for classification problem where the response variables are discrete, but also works for more generalized regression tasks where the response variables are continuous.

7.3 SECOND-STAGE FILTERING

Features with high t-statistic measure usually tend to have strong predictive power. However, the requirement for getting high t-statistic measure is sometimes too strict. A feature has to work well for all the relevant central entities to get high t-statistic measure. Sometimes a feature may not get a very high t-statistic measure but they are still effective enough for a group of central entities. To have a finer analysis of the features, we use mixed effects model to understand where its impact comes from.

Mixed effect model is a powerful tool to estimate the correlation between a response variable and some other variables that are observed along with it. It decomposes the observation into a deterministic component and a random component. Deterministic component demonstrates the detailed effects of individual objects, while the random effects are intended to explain the representative influence on the response in a more general way. For instance, suppose our task is to predict the revenue performance of movies from the movie reviews, and the movies can be clustered into two genres – drama and documentary. Drama is likely to be more popular than documentary, and gets higher revenue on average. But such kind of impact still varies a lot among individual movies in the same genre. Hence, the genres can perform as random factors, and the random effect of drama will be larger than that of documentary. Different from fixed effects, the overall random effects are assumed to sum up to zero, thus reflect the "relative " impact of the random factors when compared to each other.

Formally, denote the vector of observation as $\mathcal{Y}$, a linear mixed model can be written as:

$$\mathcal{Y} = \mathcal{X}\beta + \mathcal{U}\gamma + \epsilon \tag{7.5}$$

where $\mathcal{X}$ and $\mathcal{U}$ are the designed fix-effects and random-effects matrix respectively. β and γ are the vectors for fixed effects and random effects, and ϵ is the error. β, γ and ϵ are unknown and needs to be estimated. Moreover, the random effect and the error are assumed to be sampled from an underlying normal distribution: $\gamma \sim \mathcal{N}(0, Q)$, $\epsilon \sim \mathcal{N}(0, H)$. Different from linear regression, the primary goal of mixed effects model is not trying to minimize the error. Besides, the random effects are assumed to obey normal distribution, so the random effects are more like a rough estimation on a high level compared to fixed effects.

The problem finally boils down to minimize the following objective function when D and H are unknown [52]:

$$\mathcal{L}(V, \beta) = \ln|V| + (\mathcal{Y} - \mathcal{X}\beta)^T V^{-1}(\mathcal{Y} - \mathcal{X}\beta) + \ln|\mathcal{X}^T V \mathcal{X}| \tag{7.6}$$

where $V = \mathcal{U}Q\mathcal{U}^T + H$.

For a feature f, we assume that it affects the response variable in a deterministic way. The central entities can often be categorized into diverse clusters (e.g., based on types or other kinds of attributes), and the clustering information is used as random factors.

To decompose the impact of a feature f on the response into different components, we find all documents that contain f and denote it as D_f (see Section 7.2). D_f includes all the positive examples, which tell us how the feature makes impact on its presence. To make a full analysis of the impact of a feature on both its presence and absence, we also sample some negative examples from the data where the feature cannot be found in the documents. We denote the negative example set as D'_f and $D'_f \subset \{d|d \in D \wedge f \notin \mathcal{K}(d, E)\}$. D_f and D'_f are merged together and their indexes are recorded in Σ_f:

$$\Sigma_f = \{i|d_i \in D_f \vee d_i \in D'_f\}$$

We assume that the fixed effects come from the impact of the feature f. Let $v^{(f)}$ be the vector indicating the occurrence of feature f in the sampled documents:

$$v^{(f)} = [v^{(f)}_{i_1}, v^{(f)}_{i_2}, ..., v^{(f)}_{i_{|\Sigma_f|}}]^T$$

where $i_j \in \Sigma_f$ for $1 \leq j \leq |\Sigma_f|$ and $i_1 \leq i_2 \leq \ldots \leq i_{|\Sigma_f|}$.

$$v^{(f)}_{i_j} = \begin{cases} 1, & \text{if } f \in \mathcal{K}(d_{i_j}, E); \\ 0, & \text{otherwise.} \end{cases}$$

Then the fix-effects matrix for f can be designed as $\mathcal{X}_f = [, v^{(f)}]$ where is an all-one vector with the same dimension as $v^{(f)}$.

Suppose we have N clustering systems and the i-th system splits the central entities into m_i clusters. Let $g^{(i)}(e)$ be the function which maps an central entity e to its group label assigned by the i-th clustering system. If the i-th system categorizes e to the j-th cluster, then $g^{(i)}(e) = j$. The cluster assignment based on the i-th system can be recored in a matrix $U^{(i)}$:

$$U^{(i)} = \begin{bmatrix} U^{(i)}_{11} & U^{(i)}_{12} & \ldots & U^{(i)}_{1m_i} \\ U^{(i)}_{21} & U^{(i)}_{22} & \ldots & U^{(i)}_{2m_i} \\ \vdots & \vdots & \ddots & \vdots \\ U^{(i)}_{n1} & U^{(i)}_{n2} & \ldots & U^{(i)}_{nm_i} \end{bmatrix}$$

where

$$U_{jk}^{(i)} = \begin{cases} 1, & \text{if } g^{(i)}(\sigma(d_j)) = k; \\ 0, & \text{otherwise.} \end{cases}$$

$\sigma(d_j)$ is the function to get the corresponding central entity for document d_j: $\sigma(d_j) = e$ if $e \in E_c \wedge y_j \in e.A$

To get the random-effects matrix for feature f, we extract the sub-matrix from $U^{(i)}$ based on the sampled set Σ_f for each clustering system:

$$U_f^{(i)} = \begin{bmatrix} U_{j_1,1}^{(i)} & U_{j_1,2}^{(i)} & \dots & U_{j_1,m_i}^{(i)} \\ U_{j_2,1}^{(i)} & U_{j_2,2}^{(i)} & \dots & U_{j_2,m_i}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ U_{j_{|\Sigma_f|},1}^{(i)} & U_{j_{|\Sigma_f|},2}^{(i)} & \dots & U_{j_{|\Sigma_f|},m_i}^{(i)} \end{bmatrix}$$

where $j_k \in \Sigma_f$ for $1 \le k \le |\Sigma_f|$ and $1 \le j_1 \le j_2 \le \ldots \le j_{|\Sigma_f|} \le n$. We finally design the random-effects matrix as:

$$\mathcal{U}_f = [U_f^{(1)}, U_f^{(2)}, \ldots, U_f^{(N)}]$$

For example, if we have two documents in Σ_f and only one clustering system which assigns the corresponding central entity of the first document to the first cluster while the second one to the second cluster. Then

$$\mathcal{U}_f = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The response vector for feature f is defined as:

$$\mathcal{Y}_f = [y_{i_1}, y_{i_2}, \ldots, y_{i_{|\Sigma_f|}}]^T$$

where $i_k \in \Sigma_f$ for $1 \le k \le |\Sigma_f|$ and $1 \le i_1 \le i_2 \le \ldots \le i_{|\Sigma_f|} \le n$.

Now the fixed effects and random effects for each feature can be inferred from the mixed model:

$$\mathcal{Y}_f = \mathcal{X}_f \beta_f + \mathcal{U}_f \gamma_f + \epsilon_f$$

The fixed effects β_f contains two variables β_{f0} and β_{f1}. β_{f0} is the intercept, and β_{f1} is the deterministic effect comes from feature f. γ_f measures the random effects in a more general level. Ideally, if a feature is applicable to a wide range of central entities, the sum of random effects in γ_f is close to 0, or it could be a relatively small number compared to β_{f1}. Hence,

we estimate the deterministic impact made by feature f on the response as:

$$I(f) = \frac{|\beta_{f1}|}{|\beta_{f1}| + |\gamma_f|_1} \tag{7.7}$$

Features are first filtered by t-statistic measure and those with low t-statistic value are removed. Then deterministic impact measure based on mixed effects model (Equation 7.7) are used to further select features which are effective for a wide range of central entities. Finally a document d is represented by:

$$\cup_{l \in L} \{ f | f \in \mathcal{K}(d, E) \wedge ts(f) \geq \theta_{ts} \wedge I(f) \geq \theta_I \}$$

7.4 EXPERIMENT

In this section, we evaluate the method introduced in this chapter by using two variants of movie revenue prediction task.

7.4.1 Experimental setup

Experiment procedure and parameter setting

The task-aware knowledge graph constructed based on the generic knowledge graph built from Wiki Infobox (see Chapter 6) is used to generate the context-aware explanatory path-based features to represent the documents. Besides, many of the reviews in the dataset are short sentences which do not mention the entity name explicitly. To further enlarge the coverage, we treat explicit reference like "the movie" and "the director" as the related entity as well. They are also transformed to their relations with the central entities (e.g., "the director" is transformed to "*DirectedBy*"). There are altogether 20 such phrases which are manually selected based on 4 most prevalent relations with movie entities: *SELF*, *DirectedBy*, *WrittedBy* and *StarredBy*. Note that more complicated NLP tools can be utilized to facilitate this process as well, but we only use the most straightforward way in our experiment which still meets our primary goal of studying the effectiveness of using task-aware knowledge graph.

We compare our method with several baseline methods representing the major existing approaches for the construction of features.

- N-grams, including unigrams, bigrams and trigrams.

- N-grams with source websites: the reviews are collected from seven websites. Besides the original n-grams, we also conjoin the n-grams with the source information.
- Part-of-speech n-grams which combine the POS tags with the n-gram features.
- Dependency triples where each feature is a triple of ⟨*head word, relation, modifier word*⟩.
- Embedding vectors: a TFIDF-weighted linear combination of word vectors is used to represent documents. The word vectors are pre-trained on Common Crawl [46][1].

A stopword list containing 30 words is used. Phrases that only contains stopwords are removed, for all methods including ours.

We apply our method in two representative prediction tasks, one is to predict the weekend revenue and the other is to predict the per-screen revenue. They share the same input text but the response variables are different. L is set to be $\{1, 2, 3, 4, 5\}$ which means that we use 1-5 n-grams as contextual phrases to generate the context-aware explanatory path-based features. The maximum window size W is set to be 12. All the parameters are tuned based on the development set, resulting in the following parameter settings. T-statistic measure threshold (θ_{ts}) is set to be 503 for weekend revenue prediction, and 7.7 for the per-screen revenue prediction. Movies are clustered by genres and release date that are attributes of the movie entities, which is used as random factors in the mixed effects model. The cut-off threshold for the deterministic impact measure (θ_I) is set to be 0.30 for weekend revenue while 0.29 for per-screen revenue.

Two models are employed to predict the revenue from reviews. One is linear regression without any regularization, and the other elastic net which is linear regression with a combination of L-1 and L-2 regularization [53].

7.4.2 Evaluation results

Performance of knowledge graph-level features

The performance of our method and the baseline methods is shown in Table 7.1 and Table 7.2. "unigram", "bigram" and "trigram" are the original n-grams. "unigram+", "uni- and bigram+" and "uni-, bi- and trigram+" are n-grams conjoined with the source websites. "POS unigram", "POS uni- and bigram" and "POS uni-, bi- and trigram" use n-grams with

[1]The pre-trained vectors are downloaded from https://fasttext.cc/docs/en/crawl-vectors.html

Table 7.1: Performance of our method v.s. baseline methods with linear regression

feature	Weekend revenue		Per-screen revenue	
	MAE($M)	SMAPE (%)	MAE($K)	SMAPE (%)
unigram	9.41	70.71	7.07	38.13
uni- and bigram	9.03	68.78	6.70	33.61
uni-, bi- and trigram	9.05	68.55	6.68	33.50
unigram+	8.11	65.57	6.40	33.74
uni- and bigram+	8.16	65.52	6.32	32.70
uni-, bi- and trigram+	8.36	65.96	6.31	32.51
POS unigram	9.11	70.04	7.12	37.66
POS uni- and bigram	8.82	69.79	6.78	34.72
POS uni-, bi- and trigram	8.88	68.86	6.70	33.80
DEP triple	9.40	69.27	6.91	35.17
W2V	10.04	68.55	8.19	66.03
TAKG & unigram	8.77*	69.17	6.22*	32.55*
TAKG & uni- and bigram	8.77*	68.43	6.40*	32.17*
TAKG & uni-, bi- and trigram	8.85*	67.77	6.48*	32.62*
TAKG & unigram+	7.82*	63.94*	5.88*	31.25*
TAKG & uni- and bigram+	8.07†	65.43	6.09*	31.79*
TAKG & uni-, bi- and trigram+	8.31	65.58	6.16*	31.94*
TAKG & POS unigram	8.59*	68.79 †	6.67*	35.33*
TAKG & POS uni- and bigram	8.50*	68.26*	6.55*	32.95*
TAKG & POS uni-, bi- and trigram	8.62*	67.69*	6.60†	33.29
TAKG & DEP triple	8.82*	67.08*	6.62*	34.00*
TAKG & W2V	10.03	68.55	6.01	35.71
TAKG	**6.47***	**61.77***	**4.62***	**25.91***

Context-aware explanatory-path featurization (CaExPF) is used to represent the documents. Two-tailed t-test is done for paired data. For the last row, we compare CaExPF with all the other methods, and the minimum p-value is reported. The 13-23th row (from "CaExPF & unigram" to "CaExPF & W2V") is compared to the corresponding baseline method in the 2-12th row (e.g., "unigram" v.s. "CaExPF & unigram"). * indicates p-value < 0.01 and † indicates p-value < 0.05.

their POS tags. "DEP triple" is the dependency triples. "W2V" uses the embedding vectors. "CaExPF" is the context-aware explanatory path-based features generated by our method. "CaExPF & unigram" takes both CaExPF features and unigrams. Similarly, features like "CaExPF & unigram+" and "CaExPF & DEP triple" mean to combine CaExPF with other baseline features.

We can see that the baseline features are not very effective and suffer a lot from the overfitting problem. Taking unigram features as an example, we test the model trained for weekend revenue on a subset of the training set, and MAE is $16.54, much less than what we get for the test set (over $9 million). Elastic net can help to reduce the noise and alleviate

Table 7.2: Performance of our method v.s. baseline methods with elastic net

feature	Weekend revenue		Per-screen revenue	
	MAE($M)	SMAPE (%)	MAE($K)	SMAPE (%)
unigram	8.45	69.68	6.48	34.03
uni- and bigram	8.20	67.96	6.29	31.65
uni-, bi- and trigram	8.19	67.13	6.15	30.96
unigram+	7.87	65.02	5.98	31.65
uni- and bigram+	7.96	65.37	5.88	30.32
uni-, bi- and trigram+	7.95	65.13	5.82	29.78
POS unigram	7.93	67.91	6.92	34.81
POS uni- and bigram	7.82	66.45	6.63	33.26
POS uni-, bi- and trigram	7.85	66.07	6.49	32.74
DEP triple	9.40	69.27	6.46	32.84
W2V	8.19	66.03	6.44	32.34
TAKG & unigram	7.88*	67.53*	6.01*	31.70*
TAKG & uni- and bigram	7.96*	68.19	6.07*	30.83†
TAKG & uni-, bi- and trigram	8.01*	66.27	6.01*	30.42*
TAKG & unigram+	7.71*	64.37	5.64*	30.34*
TAKG & uni- and bigram+	7.82*	64.97	5.69*	29.58*
TAKG & uni-, bi- and trigram+	7.95	65.15	5.67*	29.14*
TAKG & POS unigram	7.64*	67.70	6.38*	33.46
TAKG & POS uni- and bigram	7.60*	66.16	6.43	32.62
TAKG & POS uni-, bi- and trigram	7.66*	65.55	6.36	32.29
TAKG & DEP triple	7.86	66.21	6.20*	31.69*
TAKG & W2V	8.17	65.65	6.43	32.33
TAKG	**6.16***	**61.59***	**4.40***	**23.58***

Context-aware explanatory-path featurization (CaExPF) is used to represent the documents. Two-tailed t-test is done for paired data. For the last row, we compare CaExPF with all the other methods, and the minimum p-value is reported. The 13-23th row (from "CaExPF & unigram" to "CaExPF & W2V") is compared to the corresponding baseline method in the 2-12th row (e.g., "unigram" v.s. "CaExPF & unigram"). * indicates p-value < 0.01 and † indicates p-value < 0.05.

the overfitting problem to some extent. The performance is improved significantly for all baseline methods, but still much worse than the CaExPF. The proposed two-stage filtering algorithm has already removed a large number of noisy features, so elastic net only produces limited improvement for CaExPF.

When we utilize NLP tools to get more sophisticated features such as part-of-speech tags and dependency triples, it sometimes works better than purely using n-gram features. When source website information is added, the n-grams features also become more powerful. Generally, source website information outperforms POS tag, and both of them works better than dependency triples. All those features are much sparser than the original n-grams,

which exacerbates the overfitting problem. Again, feature selection is needed to obtain better performance. However, the overfitting problem cannot always be fully solved with L-1 and L-2 regularization. For example, for both POS n-grams and n-grams with source websites, the prediction error is slightly increased when higher-order n-grams are included (e.g., "unigram+" is slightly better than "unigram+ & bigram+"). The embedding vectors ("W2V") does not work as well as others when the prediction is made by the linear regression model. When elastic net is employed, it can get comparable performance with other baseline methods.

CaExPF outperforms all the baseline methods significantly. Besides, when we combine the CaExPF feature with baseline features, the prediction errors are reduced compared to only using baseline features, showing that CaExPF is highly effective and quite robust. We carry out significance test to check whether the reduction is significant, and it turns out that the improvement made by adding CaExPF to baseline features is significant in general. We can further observe that using CaExPF alone works the best among all the combinations, which is a very promising result because this means that the features used in CaExPF are not only semantically interpretable and relevant to the prediction task (thus more generalizable), but also seem to be "sufficient" for our prediction tasks in the sense that once we have such effective semantic features, adding additional text-level features would tend to degrade the performance, presumably due to overfitting.

The benefits of CaExPF comes from both aspects: (1) extracting explanatory path to represent the documents along with the context, (2) noise filtering. Feature extraction helps to find informative snippets, and the transformation from entity to explanatory path makes the feature more generalizable to unseen data meanwhile reduces sparsity to avoid overfitting. To see whether the features transformation is effective, we compare the context-aware entity-based features to context-aware explanatory path-based features. We can see that CaExPF beats CAEF features on all metrics for both tasks (Table 7.3 and Table 7.4). By accurately distinguishing finer-granularity relations (i.e., the specific relation type pre-defined by the knowledge graph rather than the more general word relations such as paradigmatic relation or syntagmatic relation), the relationship between the entities mentioned in the text and the central entities is better captured by the explanatory path. With further pruning of the noise, the TAKG-based features finally work better than the KG-based features generated from the knowledge graph expanded by the word relations (i.e., paradigmatic and syntagmatic relation). In Table 7.5 and Table 7.6, we can see that the TAKG-based features filtered by the two-stage filtering method works significantly better.

Next, we investigate how effective each of the two-stage filtering steps is for generating better features, and make some further analysis of the filtering algorithm. To avoid automatic

Table 7.3: Performance of context-aware entity-based features v.s. context-aware explanatory path-based features with linear regression

feature	Weekend revenue		Per-screen revenue	
	MAE($M)	SMAPE (%)	MAE ($K)	SMAPE (%)
TAKG (CaEF)	7.49	65.36	5.75	30.27
TAKG (CaExPF)	**6.35***	**60.22***	**4.58***	**26.35***

Two-tailed t-test is done for paired data. * indicates p-value < 0.01 and † indicates p-value < 0.05.

Table 7.4: Performance of context-aware entity-based features v.s. context-aware explanatory path-based features with elastic net

feature	Weekend revenue		Per-screen revenue	
	MAE($M)	SMAPE (%)	MAE ($K)	SMAPE (%)
TAKG (CaEF)	7.13	63.80	5.60	29.16
TAKG (CaExPF)	**6.12***	**61.01***	**4.44***	**24.06***

Two-tailed t-test is done for paired data. * indicates p-value< 0.01 and † indicates p-value< 0.05.

feature selection made by regularization, we use linear regression rather than elastic net to train the model. As shown in Table 7.7, we first use all the context-aware explanatory path-based features without any filtering, and the performance is even worse than unigram because there is too much noise. Then the first-round filtering is carried out based on the t-statistic measure. Many of the noisy features are filtered out, leading to smaller prediction errors. In the second-round filtering, the mixed effects model is used to analyze the source of impact for each feature, and features that are applicable to multiple types of central entities stand out in this round. Hence, the performance is further improved.

The baseline methods we show in Table 7.1 are all automatically generated. In real applications, we can sometimes utilize manually designed features that may be more effective than text features. For example, sophisticated meta features are provided in [5]. Many of them are more than the straightforward relationship we can find in a regular knowledge graph. For example, information like number of screens, number of highest grossing actors, whether released on Labor Day, etc. We conduct experiments with the metadata and the results are shown in Table 7.8 and Table 7.9. We also combine the CaExPF with the metadata features, and the results can be found in the last row ("meta & CaExPF").

We can see that the metadata features works better than the CaExPF in the weekend revenue prediction task. When we combine them together, it achieves the best performance.

Table 7.5: Performance of TAKG features v.s. KG features by expanding KG by word relation with linear regression

feature	Weekend revenue		Per-screen revenue	
	MAE($M)	SMAPE (%)	MAE ($K)	SMAPE (%)
Wiki+$Para_{rw}$ (CaExPF)	9.84	70.40	6.80	35.24
Wiki+$Para_{sg}$ (CaExPF)	9.84	70.89	6.90	35.55
Wiki+Syn_{rw} (CaExPF)	9.89	69.00	7.03	36.76
Wiki+Syn_{sg} (CaExPF)	9.82	71.40 6.66	34.51	
Yago+$Para_{rw}$ (CaExPF)	9.73	69.44	6.94	35.78
Yago+$Para_{sg}$ (CaExPF)	9.62	69.39	6.91	35.35
Yago+Syn_{rw} (CaExPF)	9.63	67.70	6.82	34.67
Yago+Syn_{sg} (CaExPF)	9.56	69.56	7.07	35.85
Yago+Syn_{sg} (CaExPF) + unigram	8.97	70.21	6.75	34.29
TAKG (CaExPF)	**6.35***	**60.22***	**4.58***	**26.35***

Two-tailed t-test is done for paired data. * indicates p-value < 0.01 and † indicates p-value < 0.05.

Table 7.6: Performance of TAKG features v.s. KG features by expanding KG by word relation with elastic net

feature	Weekend revenue		Per-screen revenue	
	MAE($M)	SMAPE (%)	MAE ($K)	SMAPE (%)
Wiki+$Para_{rw}$ (CaExPF)	8.44	68.49	6.70	34.36
Wiki+$Para_{sg}$ (CaExPF)	8.44	67.51	6.45	33.23
Wiki+Syn_{rw} (CaExPF)	8.66	68.53	6.72	34.67
Wiki+Syn_{sg} (CaExPF)	8.60	68.77	6.34	32.22
Yago+$Para_{rw}$ (CaExPF)	8.45	68.11	6.48	33.49
Yago+$Para_{sg}$ (CaExPF)	8.26	66.53	6.55	33.41
Yago+Syn_{rw} (CaExPF)	8.43	66.96	6.37	32.63
Yago+Syn_{sg} (CaExPF)	8.27	67.54	6.80	34.36
TAKG (CaExPF)	**6.12***	**61.01***	**4.44***	**24.06***

Two-tailed t-test is done for paired data. * indicates p-value < 0.01 and † indicates p-value < 0.05.

For per-screen revenue prediction, our method still outperforms the metadata features and the reduction of prediction error is significant on all metrics. CaExPF can even beat the combination of the two. It again shows that the knowledge graph-based features are effective and sufficient, and the incorporation with elaborately-designed features does not boost the prediction accuracy but even make it worse. Besides, manual exploration of meta features is usually expensive, and it is infeasible to come up with all possibly helpful features in advance. Our knowledge graph-based method can automatically detect effective features

Table 7.7: Analysis of each step in the filtering strategy

feature	Weekend revenue		Per-screen revenue	
	MAE ($M) (Gain)	SMAPE (%) (Gain)	MAE ($K) (Gain)	SMAPE (%) (Gain)
All CaExPF features	9.62	69.06	6.71	33.97
1st-round filtering	6.85 (-2.77*)	63.95 (-5.11*)	4.98 (-1.73*)	28.87 (-5.10*)
2nd-round filtering	**6.47** (-0.38*)	**61.77** (-2.18*)	**4.62** (-0.36†)	**25.91** (-2.96*)

Linear regression without regularization is used. One-tailed t-test is performed for the gain of the performance. * indicates p-value < 0.01 and † indicates p-value < 0.05

Table 7.8: Performance of our method v.s. metadata features with linear regression

feature	Weekend revenue		Per-screen revenue	
	MAE($M)	SMAPE (%)	MAE ($K)	SMAPE (%)
meta	**6.11**	61.41	6.85	38.76
TAKG (CaExpF)	6.47	61.77	**4.62***	**25.91***
meta & TAKG (CaExpF)	**6.11**	**61.39**	6.49	35.19

Two-tailed t-test is done for paired data. In each pair, the first one is the one that get the best performance, and the second one is any of the others. * indicates p-value < 0.01 for all tests and † indicates p-value < 0.05 for all tests.

Table 7.9: Performance of our method v.s. metadata features with elastic net

feature	Weekend revenue		Per-screen revenue	
	MAE($M)	SMAPE (%)	MAE ($K)	SMAPE (%)
meta	5.62	64.81	6.43	32.74
TAKG (CaExpF)	6.16	**61.59***	**4.40***	**23.58***
meta & TAKG (CaExpF)	**5.18***	63.01	4.50	24.88

Two-tailed t-test is done for paired data. In each pair, the first one is the one that get the best performance, and the second one is any of the others. * indicates p-value < 0.01 for all tests and † indicates p-value < 0.05 for all tests.

that are both interpretable and generalizable.

Impact of the knowledge graph

Our main idea is to utilize the background knowledge provided by knowledge graph to benefit text-based prediction, which boils down to represent documents by knowledge graph-based features such as CaExPFs. The quality of the knowledge graph-based features not only depends on how the filtering strategy works (as discussed in Sectoin 6.4.1), but is also impacted by the knowledge graph itself. If the background knowledge provided by the knowledge graph is not adequate, it is hard to get sufficient high-quality knowledge graph-based features and the performance will be hurt.

To see how the knowledge graph influences the prediction task, we construct multiple versions of knowledge graph by randomly sampling entities along with their relations with the central entities. Different versions of knowledge graph can be constructed by varying the proportion of the sampling. Linear regression without regularization is used in order to keep all the knowledge graph-based features that are extracted based on the knowledge graph. The performance of the variations of task-aware knowledge graph can be found in Figure 7.1.

As the sampling is done in a random way, the quality of the knowledge graph is not necessarily higher when using larger proportion. Some variations of the original knowledge graph may contain more useful entities or relations by accident. Thus, we can observe small oscillation in all the curves. But the overall trend is still clear – with more entities and relations contained in the knowledge graph, the performance gets better. It indicates that the background knowledge provided by the knowledge graph plays an important role in the task. The knowledge graph-based features work better with more adequate background knowledge, meaning that we can expect the proposed method to become even more powerful as larger knowledge graphs of higher quality become available in the future (e.g., due to the availability of better natural language processing techniques).

7.4.3 Case study of context-aware explanatory path-based features

Our method not only brings in the existing background knowledge to solve the problem but also can tell us some interesting new underlying knowledge that can only be discovered in the data. To give an insight into how the context-aware explanatory path-based helps to explore the correlation between text and response, we show some examples of top features in Table 7.10. We find "$\langle a\ series \rangle$,$\langle SELF \rangle$,$\langle in \rangle$" is among the top features, which implies that whether a movie belongs to a movie series is helpful information for the revenue prediction. Besides, whether a movie is adapted from a graphic novel or a comic-book is correlated

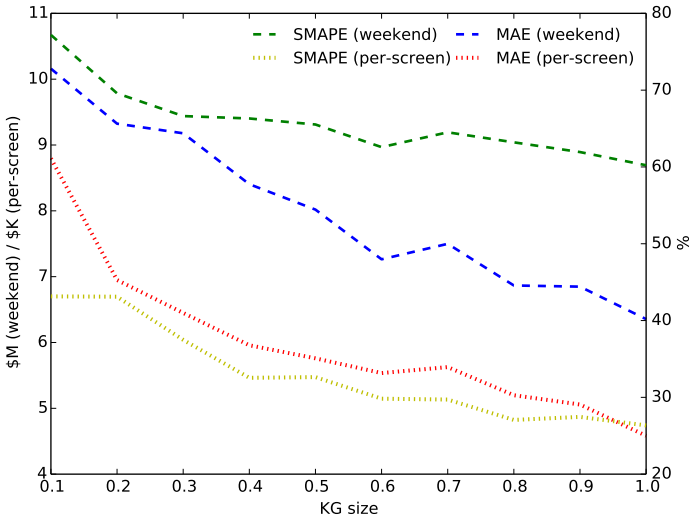


Figure 7.1: Performance of variations of KG

Table 7.10: Examples of top features

Weekend revenue	Per-screen Revenue
⟨*it* ⟩,⟨*DirectedBy*⟩,⟨*spike*⟩	⟨*first-timer*⟩, ⟨ *DirectedBy*⟩,⟨*to* ⟩
⟨*a series*⟩,⟨*SELF*⟩,⟨*in*⟩	⟨*comic-book movie*⟩,⟨*SELF*⟩,⟨⟩
⟨*like all of*⟩,⟨*WittenBy*⟩,⟨*films*⟩	⟨*toronto film festival*⟩,⟨*SELF*⟩,⟨⟩
⟨⟩,⟨*DirectedBy*⟩,⟨*charms*⟩	⟨*SELF*⟩,⟨*graphic novel*⟩,⟨⟩

to the box office performance as well. From the above three features, we can learn that if the revenue performance will be influenced by the fact that people are familiar with the background story and the fans are looking forward to watching the movie. Features like "⟨*toronto film festival*⟩,⟨*SELF*⟩,⟨⟩" implies that festival is also an important factor. People's impression of the director, e.g., whether the director is a first-timer or whether the director is charming, is also a useful feature. Although a deeper exploration of this line is out of the scope for this thesis, it is clear that the proposed context-aware explanatory path-based features also provide us a generally powerful way for potential discovery of "causal factors" that have impact on the target (response) variable.

7.4.4 Discussion of mixed effects model

Mixed effects model is a powerful and flexible statistical tool that allows for capturing correlations between response variable and potential predictor variables. In our case, mixed effects model promotes features with strong predictive ability from both global and local view. Features that are globally effective for diverse central entities are selected by the

method proposed in Section 7.3, finally leading to a further improvement of prediction accuracy on the top of the first rounding filtering. T-statistics is not normalized and the best cut-off value may vary a lot for different tasks. However, the filtering algorithm is much less sensitive to the value of deterministic impact measure (θ_I in Equation 7.7), as shown in Figure 7.2. Generally, the prediction error will be raised when the cut-off value is too small due to ineffective filtering of noise, while high cut-off value also hurts the overall performance because of aggressive pruning which misses out many good features.

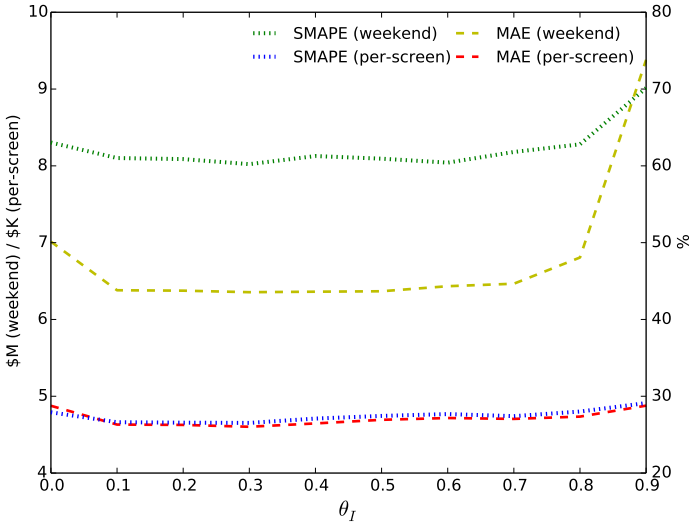


Figure 7.2: Sensitivity analysis of deterministic impact

Mixed effects model also enables us to have a deeper understanding of the correlation between the features and the response at a finer granularity. As in the example of movie revenue prediction (see Table 7.8 and Table 7.9), some features may be more effective for a certain genre than for the others. For example, the feature "⟨⟩,⟨*SELF*⟩,⟨*comic masterpieces* ⟩" contributes to romance movies more than to comedies. "⟨*first-time* ⟩,⟨*DirectedBy*⟩,⟨*with*⟩" would be a positive factor for crime movies compare to other genres. When the movie is adapted from other works such as TV shows, it is easier for action to gained high box-office.

To the best of our knowledge, our work is the first to introduce mixed effects model for feature selection. Mixed effects model is a promising tool to make exploratory analysis of data. With mixed effects model, we can dig into the effect of features at diverse level simultaneously in a more principal way, which may also benefit some downstreaming applications. For example, if we want to pay particular attention to dramas and comedies, we can quickly retrieve high-quality features by ranking them based on the random effects for the corresponding genre. It can also tell us about the interaction between different factors, such as how the release date influence certain genre of movies (if genre and release date are both

Table 7.11: Examples of random effects for features (weekend revenue prediction)

Feature	Genre	Random effect
⟨⟩,⟨*SELF*⟩,⟨*comic masterpieces* ⟩	romance	1075027.72
	comedy	130050.43
	horror	-1205078.15
⟨*first-time* ⟩,⟨*DirectedBy*⟩,⟨*with*⟩	crime	2933199.78
	drama	-63975.22
	horror	-2869224.56
⟨*film version of the* ⟩,⟨*SELF*⟩,⟨⟩	action	16146903.11
	horror	1655160.55
	drama	-3743380.12
	crime	-1301023.16
	comedy	-8400953.53
	romance	-1301859.03
	adventure	-3054847.82

used as random factors).

7.4.5 Summary

Text-based prediction has widespread applications especially for optimization of decision making. Since the target variable to be predicted is generally not directly mentioned in the text data, one of the most important challenges is how to bridge the semantic gap between the surface text content and the target variable. Overall, our experimental results clearly demonstrate that knowledge graph can be exploited to bridge the semantic gap effectively in two movie revenue prediction tasks, leading to significantly better prediction results than all the baseline methods that we experimented with.

The performance of using only knowledge graph-based features is also comparable to or sometimes even better than the manually-made features. Further analysis shows that the benefits of our method come from two aspects – making use of knowledge graph and filtering the feature in a data-driven way, which is a good sign that our knowledge graph-based method is promising to discover the underlying correlation between text input and response even when it looks weak to human beings. A case study further illustrates that with the proposed knowledge graph-based features, predictive modeling can also reveal interesting interpretable features that can help explain the changes of the response variable, suggesting its great potential as a tool for discovering and understanding causal factors that impact any interesting variable by mining text data.

Although our experiments were done on movie revenue prediction tasks, the proposed

approaches are completely general and can be easily applied to any entity-centric prediction problems and can be combined with any specific regression model to support a wide range of applications in many different domains. Further exploration of all these possibilities would be very interesting and promising future directions. Another interesting future research direction is to explore the potential of using the knowledge graph in combination with the proposed two-stage filtering algorithm to analyze causal factors behind the variation of target variables.

CHAPTER 8: SUMMARY

Text-based prediction has widespread applications especially for optimization of decision making. Since the target variable to be predicted is generally not directly mentioned in the text data, one of the most important challenges is how to bridge the semantic gap between the surface text content and the target variable. In this thesis, we propose to bridge this gap by exploiting a knowledge graph which is nowadays increasingly available in the public domain. Focusing on entity-centric tasks where the goal is to predict the attribute value of an entity, we propose a novel general algorithm for constructing knowledge-aware knowledge graph; study the knowledge graph-based feature representation; and introduce a two-stage filtering method to further reduce noisy features.

Knowledge graph can be leveraged to bridge the semantic gap and capture the correlation between the text and the response variables in general, especially when the response variables are not directly derivable from the text. However, existing applications of knowledge graphs use a static pre-constructed knowledge graph that does not make self-improvement during its application in real tasks, causing inefficiency, introduction of noise, and insufficient coverage of task-specific knowledge. We addressed this limitation by proposing to construct a task-aware knowledge graph, which bridges the gap between the construction and the application of knowledge graph, enabling dynamic optimization and adaptation of a knowledge graph in a task-specific manner.

Conceptually, if one could build a truly complete "universal knowledge graph" that would cover all the human knowledge, the task-aware knowledge graph would be simply a subgraph of such a universal knowledge graph that only contains relevant knowledge needed in a task. However, it is clearly impossible to build such a knowledge graph in practice, and the best we could achieve is realistically a general knowledge graph inevitably incomplete for specific tasks. The main idea of our proposed construction algorithms is to construct a task-aware knowledge graph based on a pre-constructed general knowledge graph adaptively for a real application task and learn new knowledge in a task-dependent way. The new knowledge learned from the task can enrich the overall cumulation of knowledge in a general knowledge graph in turn, and it can be collected through different learning processes on different data sets. For example, the aliases we extracted in our experiments for movie review classification task can be used to enrich the "synonym" relation in another general knowledge graph. If we have more data sets for other similar problems, the knowledge learned in the expansion process can also be included, leading to continuous growth of the general knowledge graph (at least conceptually).

The constructed task-aware knowledge graph captures the knowledge requirements and can provide better support for the task. As we have experimentally shown in the movie classification and revenue prediction evaluation, the task-aware knowledge can indeed help to improve the performance of the task significantly. The gain mainly comes from a better selection of features. However, it is different from the traditional feature selection techniques. Standard feature selection techniques can help to alleviate the overfitting problem, but we only observe slight improvement. This is because feature selection is data-independent and it fails when most of the features are noise such as in the case of movie classification. Task-aware knowledge graph solves the problem in a task-dependent way by leveraging relevant task knowledge to ensure generalization of the features and improve robustness to overfitting, making it possible for the features to be reused on arbitrary data sets for the same task.

Knowledge graph is an important resource to support inference as statistical learning reaches its limit. The proposed task-aware knowledge graph opens up a new way of mutual enhancing the construction and the application of knowledge graph. The instantiation is a good example of how the framework can be used in a special kind of text analysis tasks, but the framework is clearly not limited to entity-centric classification or entity-centric regression problems. An important future direction is to further explore the framework on other types of applications and make the further refinement of the framework, especially for text-based prediction tasks where the target variable to be predicted tends to be only remotely related to the text content, thus requiring knowledge graphs to help bridge the gap.

Evaluation results on two examples of entity-centric applications show that the idea of exploiting knowledge graph for text-based prediction is very powerful and that the task-aware knowledge graph is quite effective, outperforming all the baselines that use text-level features significantly and delivering promising results as compared with human-generated metadata features or human annotation performance. Our study demonstrates the effectiveness of different types of knowledge graph-based features on multiple applications. It turns out that the context information is important and the explanatory path can help to capture the correlation between the related entities mentioned in the text content and the target variables. The proposed method can also effectively reveal meaningful features that can potentially explain the variation of the target variable, making the approach also a potentially powerful tool for helping make causal discoveries by leveraging knowledge graph and text-based predictive modeling.

Although our experiments were done on movie classification and revenue prediction tasks, the proposed approaches are completely general and can be easily applied to any entity-centric prediction problems and can be combined with any specific regression model to support a wide range of applications in many different domains. Further exploration of

all these possibilities would be very interesting and promising future directions. Another interesting future research direction is to explore the potential of using the knowledge graph in combination with the proposed two-stage filtering algorithm to analyze causal factors behind the variation of target variables.

REFERENCES

[1] A. Bermingham and A. Smeaton, "On using twitter to monitor political sentiment and predict election results," in *Proceedings of the Workshop on SAAIP'11)*, 2011, pp. 2–10.

[2] M. Gaurav, A. Srivastava, A. Kumar, and S. Miller, "Leveraging candidate popularity on twitter to predict election outcome," in *Proceedings of the 7th Workshop on Social Network Mining and Analysis*. ACM, 2013, p. 7.

[3] C. Dellarocas, X. M. Zhang, and N. F. Awad, "Exploring the value of online product reviews in forecasting sales: The case of motion pictures," *Journal of Interactive marketing*, vol. 21, no. 4, pp. 23–45, 2007.

[4] A. Ghose and P. G. Ipeirotis, "Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics," *TKDE'11*, vol. 23, no. 10, pp. 1498–1512, 2011.

[5] M. Joshi, D. Das, K. Gimpel, and N. A. Smith, "Movie reviews and revenues: An experiment in text regression," in *Proceedings of NAACL HLT'10*, 2010, pp. 293–296.

[6] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[7] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," *The semantic web*, pp. 722–735, 2007.

[8] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of SIGMOD'08*, 2008, pp. 1247–1250.

[9] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of WWW'07*, 2007, pp. 697–706.

[10] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell, "Toward an architecture for never-ending language learning." in *AAAI*, vol. 5, 2010, p. 3.

[11] N. Nakashole, M. Theobald, and G. Weikum, "Scalable knowledge harvesting with high precision and high recall," in *Proceedings of WSDM'11*, 2011, pp. 227–236.

[12] F. Niu, C. Zhang, C. Ré, and J. W. Shavlik, "Deepdive: Web-scale knowledge-base construction using statistical learning and inference." *VLDS*, vol. 12, pp. 25–28, 2012.

[13] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *Proceedings of the SIGKDD'14*, 2014, pp. 601–610.

[14] M. Brambilla, S. Ceri, E. Della Valle, R. Volonterio, and F. X. Acero Salazar, "Extracting emerging knowledge from social media," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 795–804.

[15] M. Rotmensch, Y. Halpern, A. Tlimat, S. Horng, and D. Sontag, "Learning a health knowledge graph from electronic medical records," *Scientific reports*, vol. 7, no. 1, p. 5994, 2017.

[16] Y. Li, C. Liu, N. Du, W. Fan, Q. Li, J. Gao, C. Zhang, and H. Wu, "Extracting medical knowledge from crowdsourced question answering website," *IEEE Transactions on Big Data*, 2016.

[17] D. Song, F. Schilder, S. Hertz, G. Saltini, C. Smiley, P. Nivarthi, O. Hazai, D. Landau, M. Zaharkin, T. Zielund et al., "Building and querying an enterprise knowledge graph," *IEEE Transactions on Services Computing*, 2017.

[18] P. Upadhyay, T. Patra, A. Purkar, and M. Ramanath, "Teknowbase: Towards construction of a knowledge-base of technical concepts," *arXiv preprint arXiv:1612.04988*, 2016.

[19] D. Movshovitz-Attias, "Grounded knowledge bases for scientific domains," Ph.D. dissertation, Institute of General Medicines under grant number 1R01GM081293, Google, 2015.

[20] G. Meng, Y. Xue, J. K. Siow, T. Su, A. Narayanan, and Y. Liu, "Androvault: Constructing knowledge graph from millions of android apps for automated analysis," *arXiv preprint arXiv:1711.07451*, 2017.

[21] M. Rospocher, M. van Erp, P. Vossen, A. Fokkens, I. Aldabe, G. Rigau, A. Soroa, T. Ploeger, and T. Bogaard, "Building event-centric knowledge graphs from news," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 37, pp. 132–151, 2016.

[22] S. Gottschalk and E. Demidova, "Eventkg: A multilingual event-centric temporal knowledge graph," *arXiv preprint arXiv:1804.04526*, 2018.

[23] Y. Zhu, C. Zhang, C. Ré, and L. Fei-Fei, "Building a large-scale multimodal knowledge base system for answering visual queries," *arXiv preprint arXiv:1507.05670*, 2015.

[24] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs," in *ICML'17*, 2017, pp. 3462–3471.

[25] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," in *Proceedings of EMNLP'14*, 2014, pp. 615–620.

[26] S. W.-t. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," 2015.

[27] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Computational linguistics*, vol. 37, no. 2, pp. 267–307, 2011.

[28] X. Meng, F. Wei, X. Liu, M. Zhou, S. Li, and H. Wang, "Entity-centric topic-oriented opinion summarization in twitter," in *Proceedings of SIGKDD'12*, 2012, pp. 379–387.

[29] J. Dalton, L. Dietz, and J. Allan, "Entity query feature expansion using knowledge base links," in *Proceedings of SIGIR'14*, 2014, pp. 365–374.

[30] J. Guisado-Gámez, D. Dominguez-Sal, and J.-L. Larriba-Pey, "Massive query expansion by exploiting graph knowledge bases for image retrieval," in *Proceedings of ICMR'14*, 2014.

[31] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes." in *AAAI*, 2014, pp. 1112–1119.

[32] S. Zhao and R. Grishman, "Extracting relations with integrated information using kernel methods," in *Proceedings of the 43rd annual meeting on ACL'05*, 2005, pp. 419–426.

[33] L. Qian, G. Zhou, F. Kong, Q. Zhu, and P. Qian, "Exploiting constituent dependencies for tree kernel-based semantic relation extraction," in *Proceedings of the 22nd International COLING'08*, 2008, pp. 697–704.

[34] H. J. Suominen and T. I. Salakoski, "Supporting communication and decision making in finnish intensive care with language technology," *Journal of Healthcare Engineering*, vol. 1, no. 4, pp. 595–614, 2010.

[35] M. J. Paul and M. Dredze, "You are what you tweet: Analyzing twitter for public health." *Icwsm*, vol. 20, pp. 265–272, 2011.

[36] W. Zhang and S. Skiena, "Improving movie gross prediction through news analysis," in *Proceedings of WI-IAT'09*, vol. 1. IEEE, 2009, pp. 301–304.

[37] M. Schuhmacher and S. P. Ponzetto, "Knowledge-based graph document modeling," in *Proceedings of WSDM'14*, 2014, pp. 543–552.

[38] C. Wang, Y. Song, H. Li, Y. Sun, M. Zhang, and J. Han, "Distant meta-path similarities for text-based heterogeneous information networks," in *Proceedings of the CIKM'17*, 2017, pp. 1629–1638.

[39] A. Varga, A. E. C. Basave, M. Rowe, F. Ciravegna, and Y. He, "Linked knowledge sources for topic classification of microposts: A semantic graph-based approach," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 26, pp. 36–57, 2014.

[40] K. Coursey and R. Mihalcea, "Topic identification using wikipedia graph centrality," in *Proceedings of NAACL*. Association for Computational Linguistics, 2009, pp. 117–120.

[41] F. Ensan and E. Bagheri, "Document retrieval model through semantic linking," in *Proceedings of WSDM'17*, 2017, pp. 181–190.

[42] C. Xiong, R. Power, and J. Callan, "Explicit semantic ranking for academic search via knowledge graph embedding," in *Proceedings of the WWW'17*, 2017, pp. 1271–1279.

[43] E. Gabrilovich and S. Markovitch, "Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge," in *AAAI*, vol. 6, 2006, pp. 1301–1306.

[44] Z. Bitvai and T. Cohn, "Non-linear text regression with a deep convolutional neural network," in *Proceedings of ACL'15*, vol. 2, 2015, pp. 180–185.

[45] S. Massung and C. Zhai, "Non-native text analysis: A survey," *Natural Language Engineering*, vol. 22, no. 2, pp. 163–186, 2016.

[46] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[47] S. Jiang and C. Zhai, "Random walks on adjacency graphs for mining lexical relations from big text data," in *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE, 2014, pp. 549–554.

[48] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[49] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[50] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[51] S. Jiang, C. Zhai, and Q. Mei, "Exploiting knowledge graph to improve text-based prediction," in *2018 IEEE International Conference on Big Data*, 2018.

[52] B. T. West, K. B. Welch, and A. T. Galecki, *Linear mixed models: a practical guide using statistical software*. CRC Press, 2014.

[53] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.