Exploiting Knowledge Graph to Improve Text-based Prediction

Shan Jiang, Chengxiang Zhai
University of Illinois at Urbana-Champaign
sjiang18, czhai@illinois.edu

Qiaozhu Mei
University of Michigan
qmei@umich.edu

***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 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 may not be directly derivable from the text 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. In this paper, we propose to bridge this gap by using knowledge graph to construct more effective features for text representation. We propose a two-step filtering algorithm to enhance such a knowledge-aware text representation for a family of entity-centric text regression tasks where the response variable can be treated as an attribute of a group of central entities. We evaluate the proposed algorithm by using two revenue prediction tasks based on reviews. The results show that the proposed algorithm can effectively leverage knowledge graphs to construct interpretable features, leading to significant improvement of the prediction accuracy over traditional features.**

***Keywords*-Knowledge graph; text representation; text-based prediction**

I. Introduction

The rapid and continuous growth of online text data has created an unprecedented opportunity to mine big text data to discover knowledge and support many applications especially for optimization of decision making. 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 the data that they would otherwise not be able to see or easily see, thus leading to better decision making. Since humans are involved in every application domain, they would generate text data in every application domain, which we can leverage for various prediction tasks, making text-based prediction useful in virtually all the application domains. Text-based prediction can also facilitate a variety of downstreaming applications [1], [2], [3].

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.

While the formulation of a text-based prediction problem resembles 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 not explicitly mentioned in the text data and the correlation between the text input and the response variable can be very weak. That is, there is often a semantic gap between the attribute value to be predicted and the lexical features we can directly extract from the text data. For example, product reviews may be used to predict sales, but sale figures are very unlikely mentioned in the reviews.

An important technical challenge in text-based prediction is thus how to bridge the semantic gap between the text data and the target variable to be predicted. In this paper, we propose to bridge this gap by leveraging the increasingly available knowledge graphs (KGs) on the Web. The knowledge graphs can link the target variable or relevant attributes of the variable with content of the text data (e.g., entities and relations mentioned in the text), and we will show that knowledge graphs can be leveraged to construct more effective features for the prediction task.

The most popular way to represent text data 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. To be effective for prediction, deeper semantic feature representation is needed so that the underlying connection between the text input and the response variable can be better captured. For instance, if we want to predict the sales performance of products from reviews, users' feedback on individual features would be good indicators of the product quality, thus effective for the prediction task. Such a finer-grained analysis, however, would require additional knowledge on a deeper semantic layer (e.g., recognition of product features) that can not be

fully satisfied by using word-level or phrase-level features, which results in a semantic gap. The main motivation for our work is to use a knowledge graph to bridge such a gap. 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.

The generated KG-based features are not only effective to facilitate precise prediction but also interpretable and generalizable. The KG-based features are an abstraction or transformation of the surface entities, and they can be generalized to unseen data and naturally accommodate the emerging entities. Compared to intermediate features such as sentiment polarity, the KG-based features are universally applicable to different domains because 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, which is not an optimal solution because it is infeasible to exhaust all possibly useful intermediate features.

To apply KG to text-based prediction in a data-dependent way, we propose a two-step filtering approach. In the first step, a t-statistic based measure is used to select potentially better-performing features. 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 methods on two realistic scenarios of predicting revenues of unseen movies based on movie reviews, i.e., predicting the weekend revenue and predicting the per-screen revenue. The experiment results show that our methods are effective for finding good features, which can improve the prediction accuracy significantly. The proposed methods can also be used as a data mining tool to reveal interesting interpretable features that can help explain changes of values of the target variable.

II. Problem Definition

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 $Y = [y_1, y_2, \ldots, y_n]^T$ where y_i is the value of the response variable of the i-th document d_i in D. All the y's are numerical, either discrete or continuous. The text-based prediction task is thus essentially a regression task which maps D to Y: $\mathcal{F}(D \rightarrow Y)$. Entity-centric prediction is a family of text-based prediction tasks where the response variables are the attributes of a group of entities. Such kind of prediction tasks are very common, including examples such as prediction of sales of products, revenues of movies, and poll results of presidential candidates.

Entity can be considered as an ensemble of attributes with type and value. Formally, let e be an entity and $e.A$ its attribute set. An entity whose attribute value is to be predicted is called a **central entity**. If each response variable y in a prediction task is associated with an attribute of a central entity ($\forall y \in Y, |\{e|e \in E_c \wedge y \in e.A\}| = 1$), the task is an **entity-centric prediction** task.

For example, 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. 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.

III. Knowledge Graph for Text-based Prediction

In this section, we present the proposed approach for exploiting knowledge graph to improve entity-centric prediction. We first discuss how to generate KG-based features and then present a two-stage filtering algorithm to improve feature representation.

A. Generate Knowledge Graph-based Features

The first technical challenge we need to solve is how to capture the correlation between text and response which can be very weak. A starting point is to use the text-level surface features such as words or phrases, but they are (semantically) noisy, thus unable to capture the correlation. For entity-centric prediction tasks, an alternative way is to only focus on information that is related to the central entities. Text content related to the central entities is probably more informative than a random text segment. For example, if we want to predict box office for movies based on reviews, comments about the directors could be helpful. Thus our first idea in feature construction is to leverage knowledge graph to discover entities related to the central entities and use them to construct features.

1) Extract entity-level features: Given a knowledge graph, we first find entities that are closely related to the central entities, and 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

$$E = E_c \cup \{e \in E_0 | \exists e' \in E_c \ s.t. \ dis(e, e', G_0) \leq \theta_d\} \quad (1)$$

$$R = \{r \in R_0 | \exists e \in E_c, e' \in E \ s.t. \ r \in p \wedge p \in path(e, e', G_0) \wedge len(p) = dis(e, e', G_0)\} \cup \{\langle e, SELF, e\rangle | e \in E\} \quad (2)$$

where E_c are the central entitites. $path(e, e', G_0)$ 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 for feature transformation.

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. However, if we only use such entities as features, we would lose much context information about the mention of an entity. To capture the context information, we thus define our *entity-level 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). Finally, 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. The following is an example.

Example 1. *Given a sentence "Goblet of Fire is the entry in which Rowling finally took off the gloves." and an entity "Rowling", let l and W both be 2, then the set of entity-level features (denoted by $\mathcal{E}(s, e)$) has the following features:* {(⟨*"in" "which"* ⟩, ⟨ *"Rowling"* ⟩, ⟨⟩), (⟨⟩, ⟨ *"Rowling"* ⟩, ⟨ *"finally", "took"* ⟩), (⟨ *"in"* ⟩, ⟨ *"Rowling"* ⟩, ⟨ *"finally"* ⟩) , (⟨ *"in"* ⟩, ⟨ *"Rowling"* ⟩, ⟨ *"took"* ⟩), (⟨ *"which"* ⟩, ⟨ *"Rowling"* ⟩, ⟨ *"finally"* ⟩), (⟨ *"which"* ⟩, ⟨ *"Rowling"* ⟩, ⟨ *"took"* ⟩)}

Formally, the entity-level feature set for sentence s with parameters l and W is defined as:

$$\mathcal{E}(s, e, l, W) = \{(\langle w'_i, \ldots, w'_{i+l_1}\rangle, \langle w'_j, \ldots, w'_{j+k-1}\rangle, \langle w'_q, \ldots, w'_{q+l_2}\rangle) | e.n = \langle w_1, \ldots, w_k\rangle \wedge s = \langle w'_1, \ldots, w'_{|s|}\rangle \wedge w'_j = w_1 \wedge \ldots \wedge w'_{j+k-1} = w_k \wedge i + l_1 < j \wedge j - i \leq W \wedge q + l_2 \leq |s| \wedge j + k - 1 < q \wedge q + l_2 - (j + k - 1) \leq W \wedge l_1 + l_2 = l\} \quad (3)$$

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

2) Feature transformation: 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 *knowledge graph-based abstract relation features (KGARF)* 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. Building on Example 1, the following example illustrates how transformation works.

Example 2. *Given the relation between "Rowling" and the central entity "Harry Potter and the Order of the Phoenix" is "WrittenBy", the entity-level features extracted in Example 1 are transformed to $\mathcal{K}(s, e)$ =* {(⟨ *"in" "which"*⟩, ⟨*"WrittenBy"*⟩, ⟨⟩), (⟨⟩, ⟨*"WrittenBy"*⟩, ⟨ *"finally", "took"* ⟩), (⟨ *"in"* ⟩, ⟨ *"WrittenBy"* ⟩, ⟨ *"finally"* ⟩) , (⟨ *"in"* ⟩, ⟨ *"WrittenBy"* ⟩, ⟨ *"took"* ⟩), (⟨ *"which"* ⟩, ⟨ *"WrittenBy"* ⟩, ⟨ *"finally"* ⟩), (⟨ *"which"* ⟩, ⟨*"WrittenBy"*⟩, ⟨*"took"*⟩)}

Formally, the KG-based abstract relation features are defined as:

$$\mathcal{K}(s, e, l, W) = \{(\langle w'_{i+1}, \ldots, w'_{i+l_1}\rangle, \{\mathcal{T}(p) | e' \in E_c \wedge p \in path(e, e', G) \wedge path(e, e', G) = dis(e, e', G)\}, \langle w'_{q+1}, \ldots, w'_{q+l_2}\rangle) | (\langle w'_{i+1}, \ldots, w'_{i+l_1}\rangle, e.n, \langle w'_{q+1}, \ldots w'_{q+l_2}\rangle) \in \mathcal{E}(s, e, l, W)\} \quad (4)$$

Here $\mathcal{T}(p)$ is a function to get all the relation types in path p. Note that if the entity e is one of the central entities, then $\mathcal{T}(p) = \langle$"*SELF*"$\rangle$.

By setting the contextual phrase length (l in Equation 3) to different values, we can get multiple sets of KG-based features. Let L be all possible settings of contextual phrase length we want, then the KG-based abstract relation features in a document d are:

$$\mathcal{K}(d, E, W) = \cup_{s \in d, e \in E, l \in L} \mathcal{K}(s, e, l, W) \quad (5)$$

B. Primary Filtering of Features

The generated KG-based features can better capture the correlation between the text and the response variable than surface lexical features. However, the original KGARFs 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:

$$ts(f) = \frac{mean(Y_f) - mean(Y)}{\frac{std(Y_f)}{\sqrt{|Y_f|}}} \quad (6)$$

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 6 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.

C. 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 $\mathbb{Y}$, a linear mixed model can be written as:

$$\mathbb{Y} = \mathbb{X}\beta + \mathbb{U}\gamma + \epsilon \tag{7}$$

where $\mathbb{X}$ and $\mathbb{U}$ 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: $\gamma \sim \mathcal{N}(0, J)$, $\epsilon \sim \mathcal{N}(0, H)$. 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 J and H are unknown [4]:

$$\mathcal{L}(V, \beta) = \ln|V| + (\mathbb{Y} - \mathbb{X}\beta)^T V^{-1}(\mathbb{Y} - \mathbb{X}\beta) + \ln|\mathbb{X}^T V \mathbb{X}| \tag{8}$$

where $V = \mathbb{U}J\mathbb{U}^T + H$.

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 III-B). 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 $D'_f \subset \{d | d \in D \wedge f \notin \mathcal{K}(d, E, W)\}$. 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 $i_1 \leq i_2 \leq \ldots \leq i_{|\Sigma_f|}$.

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

Then the fix-effects matrix for f can be designed as $\mathbb{X}_f = [\mathbb{1}, v^{(f)}]$ where $\mathbb{1}$ 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 $U^{(i)}$:

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

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:

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

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

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

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

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

The response vector for feature f is defined as:

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

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

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

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

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:

$$I(f) = \frac{|\beta_{f1}|}{|\beta_{f1}| + |\gamma_f|_1} \tag{9}$$

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 9) is used to further select features which are effective for a wide range of central entities. Finally a document d is represented by:

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

IV. Evaluation

In this section, we evaluate the proposed method by using the task of predicting the weekend and per-screen revenue performances of movies based on their reviews.

A. Experimental Setup

1) Dataset: The dataset we used for the experiment is from a movie review and revenue dataset collected by Joshi et al. in [2]. 1718 movies released between 2005 and 2009 are collected and serve as the central entities in the task. The dataset is partitioned into training, development and test sets. Movies released between 2005 and 2007 are used for training. Movies released in 2008 and 2009 are used as development and test respectively. The central entities in the test set are all newly-emerging ones, reflecting the prediction task in a real-world application very well.

2) Knowledge graph: We build a knowledge graph by using Wikipedia infobox. Wiki pages about the movies are used. Besides reviews, the dataset provided in [2] also contains metadata such as director and genre. We use the director, author and actor/actress information in the metadata to complete the relations. The constructed knowledge graph is guaranteed to cover all the central entities. The path length restriction parameter θ_d in Equation 1 is set to be 1, which means only entities that are directly connected to the central entities are kept in the KG.

To enlarge the coverage of the knowledge graph, we expand it by finding aliases of the entity names from the dataset. First and last names are added to person entities. For each movie entity, we extract subphrases from its name. Subphrases that are composed of stop words are removed. Then the remaining ones mentioned in any of the reviews of the movie are added as its aliases. Finally, we have the 26,628 entities and 190,303 relations of 17 different types.

3) Experiment procedure and parameter setting: The knowledge graph constructed in Section IV-A2 is used to extract KGARF features. 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 references 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 KG.

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 $\langle$*head word, relation, modifier word*$\rangle$.
- Embedding vectors: a TFIDF-weighted linear combination of word vectors is used to represent documents. The word vectors are pre-trained on Common Crawl [5][1].

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

We apply our method in two 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 construct the features. The maximum window size W is set to 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 496 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.

4) Evaluation Metrics: We used two metrics to evaluate the performance, i.e., mean absolute error (MAE) and symmetric mean absolute percentage error (SMAPE). SMAPE is calculated in the following way:

$$SMAPE = \frac{1}{k} \sum_{1 \le i \le k} \frac{|y'_i - y_i|}{|y'_i| + |y_i|} \tag{10}$$

where y'_i and y are the predicted and the true response for the i-th example respectively. k is the number of test examples.

B. Evaluation Results

1) Performance of knowledge graph-level features: The performance of our method and the baseline methods is shown in Table I. "unigram," "bigram" and "trigram" are the raw n-grams. "unigram+" is unigram conjoined with the source websites. "POS unigram" combines unigrams with their POS tags. "DEP triple" is the dependency triples. "W2V" uses the embedding vectors. "KGARF" is the KG-based abstract relation features generated by our method. Features like "KGARF & unigram" mean to combine KGARF with other baseline features.

The results show that the baseline features are not very effective and suffer a lot from the overfitting problem. 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.

KGARF outperforms all the baseline methods significantly. Besides, when we combine the KGARF features with baseline features, the prediction errors are reduced compared to only using baseline features. We carry out significance test to check whether the reduction is significant, and it turns out that the improvement made by adding KGARF to baseline features is significant in general. We can further observe that using KGARF alone works the best among all the combinations, which is a very promising result because this means that the KGARF features are not only semantically

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

Table I: Performance of our method v.s. baseline methods

feature	Weekend revenue				Per-screen revenue			
	LR		Elastic net		LR		Elastic net	
	MAE($M)	SMAPE (%)	MAE ($K)	SMAPE (%)	MAE($M)	SMAPE (%)	MAE ($K)	SMAPE (%)
unigram	9.41	70.71	8.45	69.68	7.07	38.13	6.48	34.03
unigram & bigram	9.03	68.78	8.20	67.96	6.70	33.61	6.29	31.65
unigram & bigram & trigram	9.05	68.55	8.19	67.13	6.68	33.50	6.15	30.96
unigram+	8.11	65.57	7.87	65.02	6.40	33.74	5.98	31.65
unigram+ & bigram+	8.16	65.52	7.96	65.37	6.32	32.70	5.88	30.32
unigram+ & bigram+ & trigram+	8.36	65.96	7.95	65.13	6.31	32.51	5.82	29.78
POS unigram	9.11	70.04	7.93	67.91	7.12	37.66	6.92	34.81
POS unigram & bigram	8.82	69.79	7.82	66.45	6.78	34.72	6.63	33.26
POS unigram & bigram & trigram	8.88	68.86	7.85	66.07	6.70	33.80	6.49	32.74
DEP triple	9.40	69.27	8.33	67.42	6.91	35.17	6.46	32.84
W2V	10.04	68.55	8.19	66.03	8.86	46.83	6.44	32.34
KGARF & unigram	8.77*	69.17	7.88*	67.53*	6.22*	32.55*	6.01*	31.70*
KGARF & unigram & bigram	8.77*	68.43	7.96*	68.19	6.40*	32.17*	6.07*	30.83†
KGARF & unigram & bigram & trigram	8.85*	67.77	8.01*	66.27	6.48*	32.62*	6.01*	30.42*
KGARF & unigram+	7.82*	63.94*	7.71*	64.37	5.88*	31.25*	5.64*	30.34*
KGARF & unigram+ & bigram+	8.07†	65.43	7.82*	64.97	6.09*	31.79*	5.69*	29.58*
KGARF & unigram+ & bigram+ & trigram+	8.31	65.58	7.95	65.15	6.16*	31.94*	5.67*	29.14*
KGARF & POS unigram	8.59*	68.79 †	7.64*	67.70	6.67*	35.33*	6.38*	33.46
KGARF & POS unigram & bigram	8.50*	68.26*	7.60*	66.16	6.55*	32.95*	6.43	32.62
KGARF & POS unigram & bigram & trigram	8.62*	67.69*	7.66*	65.55	6.60†	33.29	6.36	32.29
KGARF & DEP triple	8.82*	67.08*	7.86	66.21	6.62*	34.00*	6.20*	31.69*
KGARF & W2V	10.03	68.55	8.17	65.65	6.01	35.71	6.43	32.33
KGARF	**6.35***	**60.22***	**6.12***	**61.01***	**4.58***	**26.35***	**4.44***	**24.06***

Two-tailed t-test is done for paired data. For the last row, we compare KGARF with all the other methods, and the minimum p-value is reported. The 13-23th row (from "KGARF & unigram" to "KGARF & W2V") is compared to the corresponding baseline method in the 2-11th row (e.g., "unigram" v.s. "KGARF & unigram"). * indicates p-value< 0.01 and † indicates p-value< 0.05.

interpretable and 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 KGARF come from two aspects: (1) feature extraction and transformation, (2) noise filtering. Feature extraction helps find informative snippets. The transformation explains the connection between the features and the central entities (thus capture the correlation with the response), and makes the features more generalizable to unseen data while also reducing sparsity to avoid overfitting. To see whether the transformation is effective, we compare the entity-level features generated from Equation 3 to KGARF. KGARF outperforms entity-level features on all metrics for both tasks (Table II), clearly indicating that the KGARF features are much more effective for the prediction task than the original entity features presumably because after the transformation, the obtained features from KG are more generalizable, effectively bridging the semantic gap.

Next, we investigate how effective each of the two-stage filtering steps is. To avoid automatic feature selection made by regularization, we use linear regression rather than elastic net to train the model. As shown in Table III, we first use all the KG-based abstract relation features without any filtering, and found that 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.

All the methods in Table I are all automatically generated from text. In some applications, we can utilize other kinds of information instead of text data. For example, we can assume that whether a movie is released on holiday is an important feature to predict the revenue. Such sophisticated meta features are provided in [2], like number of screens, number of highest grossing actors, whether released on Labor Day, etc. We compare KGARF with using metadata as features, and the results are shown in Table IV. We also combine the KGARF with the metadata features, and the results can be found in the last row ("meta & KGARF").

We can see that the performances of metadata features and KGARF are comparable in the weekend revenue prediction task. For per-screen revenue prediction, our method outperforms the metadata features and the reduction of prediction error is significant on all metrics. KGARF can even beat the combination of the two. It again shows that the KG-based features are effective and sufficient. In fact, manual exploration of meta features is expensive and labor-intensive, and it is infeasible to exhaust all possibly helpful features by hand in advance. Besides, such meta data is not applicable to other kinds of applications, or even needs to be created manually when new data comes. In comparison, our method can automatically detect effective features from text data that are both interpretable and generalizable.

2) Impact of the size of knowledge graph: The documents are represented by KG-based abstract relation features, whose quality is also impacted by the coverage of the knowledge graph. If the background knowledge is not adequate, it is hard to get sufficient high-quality KG-based

Table II: Performance of entity-level features v.s. knowledge graph-level features

feature	Weekend revenue				Per-screen revenue			
	LR		Elastic net		LR		Elastic net	
	MAE($M)	SMAPE (%)	MAE ($K)	SMAPE (%)	MAE($M)	SMAPE (%)	MAE ($K)	SMAPE (%)
Entity-level Feature	7.49	65.36	7.13	63.80	5.75	30.27	5.60	29.16
KGARF	**6.35***	**60.22***	**6.12***	**61.01***	**4.58***	**26.35***	**4.44***	**24.06***

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

Table III: 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 KGARF features	10.00	70.71	6.62	33.62
1st-round filtering	7.01 (-2.99*)	63.04 (-7.67*)	4.88 (-1.74*)	27.91 (-5.71*)
2nd-round filtering	**6.35** (-0.66*)	**60.22** (-2.82*)	**4.58** (-0.30†)	**26.35** (-1.56)

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

abstract relation features and the performance will be hurt.

To see how the coverage of the knowledge graph influences the prediction task, we construct multiple variations of the knowledge graph by randomly sampling entities along with their paths to the central entities. Linear regression without regularization is used in order to keep all the KGARF features. The performance can be found in Figure 1. As the sampling is done in a random way, the quality of the KG is not necessarily higher with larger proportion. Thus, we can observe small oscillation in the curves. But the overall trend is still clear – with larger coverage, the performance becomes better. The KGARF 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).

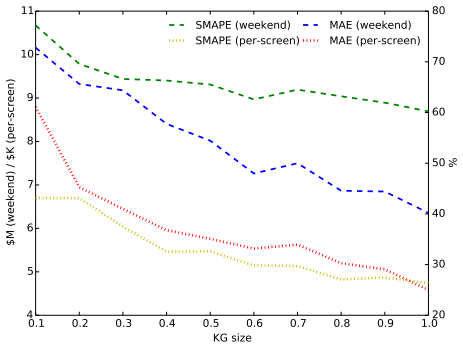

Figure 1: Performance of variations of KG

C. Case study of KG-based Abstract Relation Features

To give an insight into how the KG-based features can also help interpret the correlation between text and response, we show some examples of top features in Table V. Whether a movie belongs to a movie series, whether a movie is adapted from a graphic novel or a comic-book are all highly-ranked features. It means that 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. Festival is also an important factor, and people's impression of the director also affects the revenue performance. Although a deeper exploration of this line is out of the scope for this paper, it is clear that the proposed KG-based abstract relation features also provide us a generally powerful way for potential discovery of "causal factors" that have impact on the target variable.

D. Discussion of Mixed Effects Model

T-statistics is not normalized and the best cut-off value may vary a lot for different tasks. However, the mixed effects model-based filtering is less sensitive to cut-off threshold (θ_I in Equation 9), as shown in Figure 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.

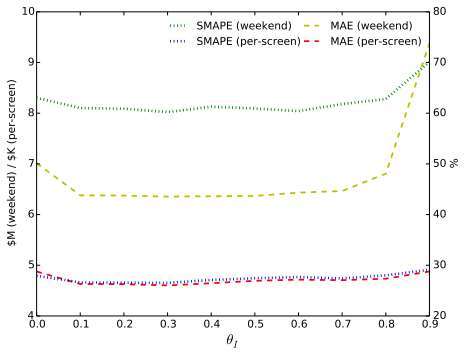

Figure 2: Sensitivity analysis of deterministic impact

Mixed effects model also enables us to have a deeper understanding of the correlation between feature and response at a finer granularity. As in the example of movie revenue prediction (Table IV), some features are more effective for a certain genre than for the others. For example, "$\langle$*comic masterpiece*$\rangle$,$\langle$*SELF*$\rangle$,$\langle\rangle$" contributes to romance movies more than to horror. "$\langle$*first-time*$\rangle$,$\langle$*DirectedBy*$\rangle$,$\langle$*with*$\rangle$" 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 musicals and drama to gain high box-office compared to mystery and documentary.

To the best of our knowledge, our work is the first to introduce mixed effects model for feature selection. Mixed effects model is a powerful tool to make exploratory analysis

Table IV: Text signals v.s. metadata features

feature	Weekend revenue				Per-screen revenue			
	LR		Elastic net		LR		Elastic net	
	MAE($M)	SMAPE (%)	MAE ($K)	SMAPE (%)	MAE($M)	SMAPE (%)	MAE ($K)	SMAPE (%)
meta	**6.11**	61.41	5.62	64.81	6.85	38.76	6.43	32.74
KGARF	6.35	**60.22**	6.12	**61.01**	**4.58***	**26.35***	**4.44**	**24.06**
meta & KGARF	**6.11**	61.39	**5.18**	63.01	6.49	35.19	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.

Table V: 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*⟩,⟨⟩

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

Feature	Genre	Random effect
⟨*comic masterpiece* ⟩,⟨*SELF*⟩,⟨⟩	romance	2319.56
	drama	626.79
	comedy	0
	horror	-626.79
	foreign	-2319.56
⟨*first-time* ⟩,⟨*DirectedBy*⟩,⟨*with*⟩	crime	7001.47
	drama	-1082.63
	comedy	-5918.83
⟨*film version of the* ⟩,⟨*SELF*⟩,⟨⟩	musical	46786.62
	drama	17041.02
	romance	-9415.62
	comedy	-14791.90
	mystery	-14953.77
	documentary	-24666.41

of data, which helps us to dig into the effect of features at diverse level simultaneously in a principal way. It can 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.

V. Related Work

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]), health research ([6], [7]) and product sales prediction ([2], [3]). However, the existing methods have not fully exploited knowledge graph to construct effective and interpretable features for prediction, which is a main contribution of our work.

Due to the semantic gap between surface form of text and the response variable in a complex prediction task, direct prediction of a variable based on text data is very challenging; as a result, semi-structured text is more popularly used. For example, hashtag and user information are useful features when leveraging tweets [1]. 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 [8]. 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 the 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 [9], [10], topic discovery [11], [12], opinion mining [13], [14] and information retrieval [15], [16]. Knowledge graphs 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. [17]) or sub-graph level (e.g. [9], [15]). In this paper, 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 integration of local context with background knowledge.

The proposed method mainly focuses on finding better features for entity-centric text regression, which is orthogonal to the choice of regression models and thus can be potentially combined with any improvement of regression models (e.g., the non-linear model proposed in [18]) to achieve a positive additive effect. While feature selection (filtering) has been well studied for classification task, it is less studied for regression. The proposed two-stage filtering algorithm is a general approach that can be potentially applied to other application scenarios for feature filtering.

VI. Conclusion

Text-based prediction has widespread applications. Since the target variable to be predicted is generally not directly mentioned in the text data, effective prediction requires bridging this semantic gap. We propose to exploit the increasingly available knowledge graphs in the public domain to bridge this gap. 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" feature representation, which is not only more interpretable, but also generalizing better than the surface text-based features. We further propose a two-stage feature filtering strategy to improve the quality of features. Evaluation results on two variants of the movie revenue prediction task show that the idea of exploiting knowledge graph for text-based prediction is very powerful, outperforming all the baselines that use text-level features significantly. Our method, which is fully automatic, even delivers comparable or better results as compared with human-generated metadata features. The results also show that both stages of filtering are necessary and that the proposed feature transformation method is effective to generate interpretable and generalizable features that perform significantly better than the original entity-level features.

Our work also demonstrates that knowledge graph provides an excellent framework for humans to inject their expertise into the process of constructing effective features (via curating/improving a knowledge graph), thus helping us naturally optimize the collaboration of humans and a machine learning system in real-world applications.

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

VII. Acknowledgement

We would like to acknowledge Professor Jingchen Liu, Department of Statistics at Columbia University, for enlightening discussions on mixed effects model. This work is supported in part by the National Science Foundation under grant numbers IIS-1633370, SMA-1620319, CNS-1513939, and CNS-1801652.

References

[1] M. Gaurav, A. Srivastava, A. Kumar, and S. Miller, "Leveraging candidate popularity on twitter to predict election outcome," in *Workshop on SNAM'13*. ACM, 2013, p. 7.

[2] 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.

[3] 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.

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

[5] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *Proceedings of LREC'18*, 2018.

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.