Hyperlink Classification via Structured Graph Embedding

Geon Lee
Sungkyunkwan University (SKKU)
Suwon, Korea
geonlee@skku.edu

Seonggoo Kang
Naver Corporation
Seongnam, Korea
seong.goo.kang@navercorp.com

Joyce Jiyoung Whang*
Sungkyunkwan University (SKKU)
Suwon, Korea
jjwhang@skku.edu

ABSTRACT

We formally define a hyperlink classification problem in web search by classifying hyperlinks into three classes based on their roles: navigation, suggestion, and action. Real-world web graph datasets are generated for this task. We approach the hyperlink classification problem from a structured graph embedding perspective, and show that we can solve the problem by modifying the recently proposed knowledge graph embedding techniques. The key idea of our modification is to introduce a relation perturbation while the original knowledge graph embedding models only corrupt entities when generating negative triplets in training. To the best of our knowledge, this is the first study to apply the knowledge graph embedding idea to the hyperlink classification problem. We show that our model significantly outperforms the original knowledge graph embedding models in classifying hyperlinks on web graphs.

CCS CONCEPTS

• **Information systems** → *Web indexing*;

KEYWORDS

Web; Hyperlink; Classification; Embedding; Graph.

ACM Reference Format:
Geon Lee, Seonggoo Kang, and Joyce Jiyoung Whang. 2019. Hyperlink Classification via Structured Graph Embedding. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331325

1 INTRODUCTION

On a web graph, a node indicates a web page and a directed edge indicates a hyperlink between the web pages. The hyperlinks are created for different reasons, and thus, may play different roles in the graph. For example, some hyperlinks are designed to navigate the main website, e.g., 'go home' or 'go back' links (*navigation* links). Some hyperlinks are made to invoke actions such as 'edit', 'share', or 'send an email' (*action* links). Some hyperlinks suggest users to take a look at related and useful information (*suggestion* links). As an example, on Stack Overflow, some people recommend a questioner to read a related page, e.g., 'see this page' by providing the URL of the page. Since these three different types of hyperlinks[1], i.e., *navigation*, *suggestion*, and *action*, imply semantically different roles, it is beneficial to classify the hyperlinks based on these roles. For example, when we compute PageRank [6] or run the HITS algorithm [9], it is preferred that the *suggestion* links are mainly taken into account. On the other hand, the *navigation* links are important when we want to trace a set of main pages of a website. Once we correctly classify hyperlinks based on their purposes, we can improve the performance of various web mining tasks.

One of the challenges of the hyperlink classification problem is that the *navigation* links are prevalent while there are very few *suggestion* and *action* links. On this imbalanced classification problem, a neighborhood-based approach such as [1] fails to correctly predict the class labels by assigning all the hyperlinks to the *navigation* class. To detect the *navigation* links, boilerplate detection has been studied [10] and an entropy-based analysis has been also considered [8]. However, these methods require rich information about the web pages and involve complicated heuristics.

We propose applying a knowledge graph embedding idea to the hyperlink classification problem. To the best of our knowledge, our work is the first study to approach the hyperlink classification problem from a structured graph embedding perspective. We generate three real-world web graph datasets by web crawling and assigning class labels to the hyperlinks. By analyzing these real-world graphs, we find that the three different types of hyperlinks are not randomly organized but preserve a characterized structure, which enables us to classify hyperlinks based on link analysis. We show that the link structure can be effectively captured via knowledge graph embedding techniques.

Knowledge graph embedding methods are different from general graph embedding techniques in that the general graph embedding methods, e.g., `node2vec` [7] and `struc2vec` [12], mainly consider the connectivity structure of a graph and only focus on representing the nodes in a low-dimensional feature space while knowledge graph embedding models aim to embed the relations as well as the entities in a feature space. We modify the recently proposed knowledge graph embedding methods, `TransE` [4], `TransH` [15], and `TransR` [11] to appropriately adapt these methods to the hyperlink classification problem. The key idea of our modification is to introduce a relation perturbation while the original knowledge graph embedding models only corrupt entities when generating negative triplets in training. This modification plays a critical role in boosting the performance of the classification model, which results in significantly outperforming the original knowledge graph embedding models in classifying hyperlinks on web graphs.

*Corresponding Author.

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331325

[1] One might insist that there should be more than three classes when we classify the hyperlinks. Even though we focus on the three-class problem in this paper, we believe that our study can be extended to the case where there are more than three classes.

2 KNOWLEDGE GRAPH EMBEDDING

A knowledge graph has been recognized as a reasonable model to encode human knowledge [3]. Given a set of known facts which can be described by a set of triplets such as (a head entity h, relation r, a tail entity t), we can create a directed graph where a node indicates an entity and a directed edge indicates a relation between the entities. Knowledge graph embedding techniques [5] have recently gained considerable attention where the goal is to represent the entities and relations in a feature space while preserving the structure of the graph [13]. Among the knowledge graph embedding methods, `TransE` [4], `TransH` [15], and `TransR` [11] are well-known methods [14]. The basic idea of these methods is to find a feature vector $\boldsymbol{h} \in \mathbb{R}^k$ of an entity h, $\boldsymbol{t} \in \mathbb{R}^k$ of an entity t, and an embedding of $\boldsymbol{r} \in \mathbb{R}^d$ of a relation r where k and d are the dimensions of the corresponding feature spaces ($k = d$ or $k \neq d$ depending on the model). Given a set of known facts (i.e., *golden triplets*) denoted by $\mathcal{S}$ and a set of *corrupted triplets* (or *negative triplets*) $\mathcal{S}'$, all the `TransE`, `TransH`, and `TransR` methods minimize the following loss function:

$$L = \sum_{(h,r,t)\in\mathcal{S}} \sum_{(h',r,t')\in\mathcal{S}'} [f(h,r,t) + \gamma - f(h',r,t')]_+ \tag{1}$$

where $[x]_+ \equiv max(0,x)$ and γ is the margin. How to create the corrupted triplets $\mathcal{S}'$ is an important issue which is differently handled depending on the model. Although the details about the corruption process vary, all the `TransE`, `TransH`, and `TransR` methods only corrupt entities when generating the negative triplets. That is, given $(h,r,t) \in \mathcal{S}$, a negative triplet is created by (h',r,t').

In (1), the way how $f(h,r,t)$ is computed determines the three different models. In `TransE`, $f(h,r,t)$ is defined to be

$$\texttt{TransE:} \quad f(h,r,t) = \|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|_2^2. \tag{2}$$

While `TransE` assumes $\boldsymbol{h}+\boldsymbol{r} \approx \boldsymbol{t}$, it has been noticed that the `TransE` model fails to correctly represent reflexive or one-to-many/many-to-one/many-to-many relations [15]. To overcome this problem, `TransH` is proposed by defining $f(h,r,t)$ to be

$$\texttt{TransH:} \quad f(h,r,t) = \|\boldsymbol{h}_\perp + \boldsymbol{r} - \boldsymbol{t}_\perp\|_2^2 \tag{3}$$

where $\boldsymbol{h}_\perp$ and $\boldsymbol{t}_\perp$ represent projected entities on a relation-specific hyperplane $\boldsymbol{w}_r$. Note that $\boldsymbol{h}_\perp = \boldsymbol{h} - \boldsymbol{w}_r{}^T\boldsymbol{h}\boldsymbol{w}_r$ and $\boldsymbol{t}_\perp = \boldsymbol{t} - \boldsymbol{w}_r{}^T\boldsymbol{t}\boldsymbol{w}_r$. Instead of representing entities and relations in the same feature space, `TransR` embeds entities and relations in distinct spaces by introducing a projection matrix $\boldsymbol{M}_r \in \mathbb{R}^{k\times d}$ which projects entities to a relation space. Then, $f(h,r,t)$ is defined to be

$$\texttt{TransR:} \quad f(h,r,t) = \|\boldsymbol{h}_r + \boldsymbol{r} - \boldsymbol{t}_r\|_2^2 \tag{4}$$

where $\boldsymbol{h}_r = \boldsymbol{h}\boldsymbol{M}_r$ and $\boldsymbol{t}_r = \boldsymbol{t}\boldsymbol{M}_r$.

The loss function defined in (1) is minimized using stochastic gradient descent with a mini-batch mode while appropriate normalizations are performed depending on the model.

3 HYPERLINK CLASSIFICATION MODEL

We create three real-world web graphs for the hyperlink classification problem[2], and propose a hyperlink classification model by modifying a knowledge graph embedding method.

[2]The datasets and the detailed descriptions about the datasets are available on http://bigdata.cs.skku.edu.

Table 1: Real-World Web Graphs. The numbers of *navigation*, *suggestion*, and *action* hyperlinks are shown.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	*navigation*	*suggestion*	*action*
`web_437`	404	437	268 (61.33%)	112 (25.63%)	57 (13.04%)
`web_1442`	332	1,442	1,284 (89.04%)	93 (6.45%)	65 (4.51%)
`web_10000`	2,202	10,000	9,892 (98.92%)	85 (0.85%)	23 (0.23 %)

3.1 Real-World Web Graphs

We create three real-world web graphs by crawling a set of web pages and the hyperlinks starting from a web page in Stack Overflow. From the seed, we randomly sample outgoing and incoming hyperlinks of the page so that we expand the seed to its direct neighbors. From the seed set, we conduct a biased random walk to sample the graph around the seeds. Table 1 shows the three datasets: `web_437`, `web_1442`, and `web_10000`.

When we create `web_437`, we give more chances to follow a non-navigational link to balance the number of hyperlinks in each class. For `web_1442`, we do not assign prior bias on the hyperlinks when conducting the random walk while we apply some heuristics to filter out the trivially removable *navigation* hyperlinks. On `web_10000`, we do not apply any tricks to remove the *navigation* hyperlinks. Thus, the distribution of the *navigation*, *suggestion*, and *action* hyperlinks on this dataset may be close to the underlying distribution of the hyperlinks in an entire web graph.

Three senior engineers in NAVER have manually labeled the hyperlinks based on consistent criteria. The assigned labels are cross checked. On the largest graph, `web_10000`, some labels are mechanically assigned by exploiting the template of a web page.

3.2 Model Specification and Training

Given a directed web graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{p_1, p_2, \cdots, p_n\}$ and $\mathcal{E} = \{(p_i, p_j) : p_i \in \mathcal{V}, p_j \in \mathcal{V}\}$, each hyperlink belongs to one of the three classes: *navigation*, *suggestion*, and *action*. If we consider a web page to be an entity, a labeled directed edge from a page to another can be thought of as a relation between the entities in a knowledge graph. Let us define three different relation labels $\mathcal{R} = \{n, s, a\}$ where n indicates *navigation*, s indicates *suggestion*, and a indicates *action*. Assigning the relation labels to the hyperlinks leads to a set of golden triplets $\mathcal{S} = \{(p_i, r, p_j) : p_i \in \mathcal{V}, r \in \mathcal{R}, p_j \in \mathcal{V}\}$. Then, we can compute an embedding of each relation as well as a set of embeddings for the web pages by minimizing the following loss function.

$$L = \sum_{(p_i,r,p_j)\in\mathcal{S}} [f(p_i,r,p_j) + \gamma - f(\mathtt{c}(p_i,r,p_j))]_+ \tag{5}$$

where $\mathtt{c}(p_i,r,p_j)$ is defined by

$$\mathtt{c}(p_i,r,p_j) = \begin{cases} \text{prob. } \alpha/2: & (p_i,r,q), q \in \mathcal{V}\backslash\{p_j\}, (p_i,r,q) \notin \mathcal{S} \\ \text{prob. } \alpha/2: & (q,r,p_j), q \in \mathcal{V}\backslash\{p_i\}, (q,r,p_j) \notin \mathcal{S} \\ \text{prob. } (1-\alpha): & (p_i,r',p_j), r' \in \mathcal{R}\backslash\{r\} \end{cases} \tag{6}$$

where α controls the chance to corrupt entities and $0 < \alpha \leq 1$. Note that if we set $\alpha = 1$, we can make the above loss function identical to that of the `TransE`, `TransH`, and `TransR` models by assigning appropriate prior probability on corrupting p_i or p_j and using the corresponding distance function $f(p_i,r,p_j)$ discussed in Section 2.

It is important to notice that $\mathcal{S}$ is a training set. For each golden triplet $(p_i, r, p_j) \in \mathcal{S}$, we generate a corrupted triplet $c(p_i, r, p_j)$ by (6). Then, we minimize (5) by incorporating all the golden triplets and the corrupted triplets, i.e., the embeddings are trained in a way that the golden triplets are encouraged and the corrupted triplets are discouraged. Once training is done, we predict relation labels for a test set $\mathcal{T} = \{(p_i, r, p_j) : p_i \in \mathcal{V}, r \in \mathcal{R}, p_j \in \mathcal{V}, (p_i, r, p_j) \notin \mathcal{S}\}$.

The difference between (1) and (5) is how to generate the corrupted triplets. As discussed in Section 2, the previously studied knowledge graph embedding methods only corrupt the entities and do not corrupt the relations. However, when we corrupt the entities, there is a chance that the corrupted triplet is not a *corrupted* one but just an *unobserved* one in the training set. For example, if we corrupt a golden triplet (p_1, n, p_2) by randomly replacing the tail of the golden triplet and make (p_1, n, p_3) which is considered to be a corrupted triplet, there is a risk that $(p_1, n, p_3) \in \mathcal{T}$ whereas $(p_1, n, p_3) \notin \mathcal{S}$. Therefore, using (p_1, n, p_3) as a negative triplet might mislead the training. Indeed, in our hyperlink classification problem, for the *navigation* relation, it is not desirable to corrupt the entities to create negative triplets because it is likely that the corrupted triplet exists in a test set $\mathcal{T}$ due to the fact that there are many *navigation* hyperlinks in the dataset as discussed in Section 3.1.

On the other hand, if we corrupt a relation, it is guaranteed that the corrupted triplet is not in a test set because each pair of the entities has a unique relation. That is, if (p_1, n, p_2) is observed, then (p_1, s, p_2) and (p_1, a, p_2) should not hold. Thus, it is safe to use (p_1, s, p_2) or (p_1, a, p_2) as corrupted triplets. However, if we only corrupt relations and do not corrupt entities to create the negative triplets, we might have a overfitting problem and the model is not sufficiently trained for an unobserved entity. Therefore, in (6), we corrupt the entities with the probability α, and corrupt the relations with the probability $1-\alpha$. When we corrupt the entities, we replace either the head or the tail with the same probability. We observe that this new corruption strategy plays a critical role in improving the model performance for the hyperlink classification problem.

3.3 Prediction

After training, we get a set of feature vectors for the web pages, denoted by $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n\}$ and a set of feature vectors for the three relations, *navigation*, *suggestion*, and *action*. Let $\boldsymbol{r}$ denote a feature vector representation of a relation r. These embeddings are computed by minimizing (5).

In testing phase, for a directed edge $(p_i, p_j) \in \mathcal{T}$, we predict the relation $r \in \mathcal{R}$ for (p_i, p_j) by computing

$$r^* = \underset{r \in R}{argmin}\ f(p_i, r, p_j) \tag{7}$$

where r^* is the predicted relation. The distance function $f(p_i, r, p_j)$ is differently defined depending on the TransE, TransH, and TransR models as discussed in Section 2. For example, if we use TransH,

$$f(p_i, r, p_j) = \|(\boldsymbol{p}_i - \boldsymbol{w}_r{}^T \boldsymbol{p}_i \boldsymbol{w}_r) + \boldsymbol{r} - (\boldsymbol{p}_j - \boldsymbol{w}_r{}^T \boldsymbol{p}_j \boldsymbol{w}_r)\|_2^2 \tag{8}$$

where $\boldsymbol{w}_r$ is a trained relation-specific hyperplane. That is, we represent the web pages (p_i, p_j) and the relations in the embedded spaces, and then assign a relation to (p_i, p_j) by taking the relation that yields the smallest distance.

Table 2: The average F1 scores (%) of our model with different α values and the original TransE, TransH, and TransR.

		TransE	TransH	TransR
web_437	Our model, α = 0.3	34.29	**60.25**	57.99
	Our model, α = 0.5	34.39	58.87	57.32
	Our model, α = 0.7	33.88	58.91	**59.83**
	The original model	**36.22**	54.04	53.22
web_1442	Our model, α = 0.3	23.39	53.42	**50.04**
	Our model, α = 0.5	**24.86**	**55.16**	46.18
	Our model, α = 0.7	21.18	52.70	45.12
	The original model	20.05	29.94	10.35
web_10000	Our model, α = 0.3	**20.68**	**76.00**	**53.86**
	Our model, α = 0.5	17.98	74.64	46.99
	Our model, α = 0.7	19.50	72.94	44.11
	The original model	15.31	25.35	2.08

Table 3: F1 score (%) of each class and the average F1 score. Our model achieves the highest F1 scores.

		navigation	*suggestion*	*action*	Average
web_437	Random-predict	59.75	25.81	11.07	32.21
	Rule-based	60.20	20.96	0.00	27.05
	TransE-original	55.78	31.96	20.93	36.22
	TransH-original	70.80	52.75	38.56	54.04
	TransR-original	67.87	52.86	38.94	53.22
	Our Model	**77.04**	**57.05**	**46.64**	**60.25**
web_1442	Random-predict	89.13	5.18	5.65	33.32
	Rule-based	72.98	10.20	36.67	39.95
	TransE-original	42.54	8.57	9.05	20.05
	TransH-original	54.80	13.57	21.45	29.94
	TransR-original	0.00	12.97	18.09	10.35
	Our Model	**93.48**	**22.88**	**49.12**	**55.16**
web_10000	Random-predict	98.91	1.60	0.00	33.50
	Rule-based	68.81	1.74	9.92	26.82
	TransE-original	43.25	2.06	0.61	15.31
	TransH-original	63.01	12.02	1.03	25.35
	TransR-original	0.00	5.61	0.61	2.08
	Our Model	**99.66**	**83.22**	**45.12**	**76.00**

4 EXPERIMENTAL RESULTS

We test the performance of our model and the original knowledge graph embedding methods[3] on the datasets discussed in Section 3.1. We compute the average F1 score by averaging the F1 score of each class. Table 2 shows the average F1 scores of our model with different α values and the performance of the original TransE, TransH, and TransR. We first observe that TransH tends to show better performance than TransE and TransR. More importantly, our model significantly outperforms the original knowledge graph embedding methods. The strategy of creating corrupted triplets plays a critical role in the hyperlink classification problem, and our strategy is effective enough to boost the performance of the original knowledge graph embedding methods.

Table 3 and Figure 1 show the F1 score of each class, and the average F1, precision, and recall scores. 'Random-predict' indicates the performance of random prediction while preserving the number of

[3]For the original knowledge graph embedding models, we use the codes from https://github.com/thunlp/KB2E. We conduct 5-fold cross validation.

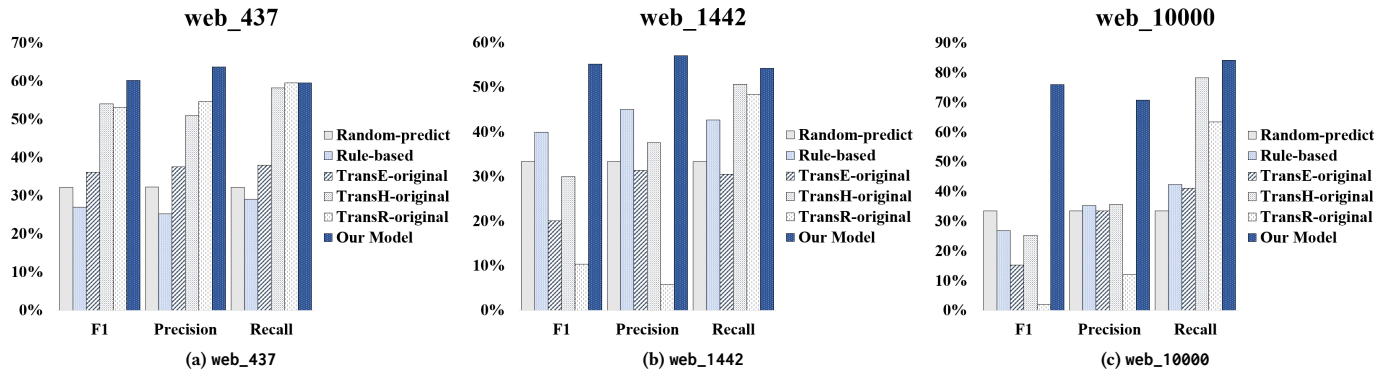

Figure 1: The average F1, average precision, and average recall on the three web graphs. Our model shows the best performance.

Table 4: Performance on the original web graphs and the randomly shuffled graphs where the relation labels are randomly assigned. The real-world web graphs have characterized structures in terms of forming each relation type.

		navigation			*suggestion*			*action*		
		F1 (%)	Pre. (%)	Rec. (%)	F1 (%)	Pre. (%)	Rec. (%)	F1 (%)	Pre. (%)	Rec. (%)
web_437	Original Graph	77.04	78.82	75.37	57.05	50.43	65.77	46.64	62.00	37.43
	Randomly Shuffled Graph	58.60	60.51	56.88	25.36	24.39	26.59	13.79	13.26	14.42
web_1442	Original Graph	93.48	92.22	94.78	22.88	30.66	18.28	49.12	48.52	49.74
	Randomly Shuffled Graph	86.08	88.94	83.41	6.19	5.28	7.53	5.68	4.58	7.52
web_10000	Original Graph	99.66	99.82	99.50	83.22	77.84	89.41	45.12	34.91	63.77
	Randomly Shuffled Graph	98.43	98.94	97.92	1.28	0.99	1.83	0.61	0.38	1.45

hyperlinks in each class. We also compare with a rule-based prediction (denoted by 'Rule-based') where we consider within-domain hyperlinks to be navigational links, the hyperlinks associated with an anchor text containing 'edit', 'share', 'email', or 'vote' to be action links, and the rest to be suggestion links. For 'Our Model', we use the result of TransH with $\alpha = 0.3$, $\alpha = 0.5$, and $\alpha = 0.3$ for web_437, web_1442, and web_10000, respectively. We see that our model achieves the best performance in terms of all the metrics.

To analyze why our approach works well for the hyperlink classification problem, we generate randomly shuffled graphs where the relation labels are randomly shuffled while preserving the number of hyperlinks in each relation. Table 4 shows the results of our model on the original graphs and the randomly shuffled graphs. We see that the classification performance significantly degrades on the randomly shuffled graphs. This shows that a web graph preserves a characterized structure with respect to the three different types of hyperlinks, which enables us to predict the relation labels via structured graph embedding.

5 CONCLUSIONS & FUTURE WORK

By introducing effective strategies for creating a set of corrupted triplets to a knowledge graph embedding method, we are able to successfully classify hyperlinks on web graphs. We plan to extend our analysis to a case where we can incorporate various features or attributes of web pages or the hyperlinks [2], and exploit clustering structure of a web graph [16].

ACKNOWLEDGMENTS

This research was supported by NAVER Corp., National Research Foundation of Korea funded by MSIT(2019R1C1C1008956, 2018R1A5A1059921), and IITP grant funded by MSIT (2019-0-00421, AI Graduate School Support Program). J. Whang is the corresponding author.

REFERENCES

[1] C. Aggarwal, G. He, and P. Zhao. 2016. Edge classification in networks. In *ICDE*.
[2] C. Aggarwal, Y. Li, P. S. Yu, and Y. Zhao. 2017. On Edge Classification in Networks with Structure and Content. In *ICDE*.
[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*. 1247–1250.
[4] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*.
[5] A. Bordes, J. Weston, R. Collobert, and Y. Bengio. 2011. Learning Structured Embeddings of Knowledge Bases. In *AAAI*.
[6] S. Brin and L. Page. 1998. The Anatomy of a Large-scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems* 30, 1-7 (1998).
[7] A. Grover and J. Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*.
[8] H.-Y. Kao, S.-H. Lin, J.-M. Ho, and M.-S. Chen. 2004. Mining Web Informative Structures and Contents Based on Entropy Analysis. *TKDE* 16, 1 (2004).
[9] J. Kleinberg. 1999. Authoritative Sources in a Hyperlinked Environment. *J. ACM* 46, 5 (1999), 604–632.
[10] C. Kohlschütter, P. Fankhauser, and W. Nejdl. 2010. Boilerplate Detection using Shallow Text Features. In *WSDM*.
[11] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*.
[12] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo. 2017. struc2vec: Learning Node Representations from Structural Identity. In *KDD*.
[13] C.-C. Wang and P.-J. Cheng. 2018. Translating Representations of Knowledge Graphs with Neighbors. In *SIGIR*.
[14] Q. Wang, Z. Mao, B. Wang, and L. Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *TKDE* 29, 12 (2017).
[15] Z. Wang, J. Zhang, J. Feng, and Z. Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes.. In *AAAI*.
[16] J. J. Whang and I. S. Dhillon. 2017. Non-Exhaustive, Overlapping Co-Clustering. In *CIKM*. 2367–2370.