Knowledge-Driven Event Embedding for Stock Prediction

Xiao Ding[†]*, **Yue Zhang**[‡], **Ting Liu**[†], **Junwen Duan**[†]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{xding, tliu, jwduan}@ir.hit.edu.cn
[‡]Singapore University of Technology and Design
yue_zhang@sutd.edu.sg

Abstract

Representing structured events as vectors in continuous space offers a new way for defining dense features for natural language processing (NLP) applications. Prior work has proposed effective methods to learn event representations that can capture syntactic and semantic information over text corpus, demonstrating their effectiveness for downstream tasks such as event-driven stock prediction. On the other hand, events extracted from raw texts do not contain background knowledge on entities and relations that they are mentioned. To address this issue, this paper proposes to leverage extra information from knowledge graph, which provides ground truth such as attributes and properties of entities and encodes valuable relations between entities. Specifically, we propose a joint model to combine knowledge graph information into the objective function of an event embedding learning model. Experiments on event similarity and stock market prediction show that our model is more capable of obtaining better event embeddings and making more accurate prediction on stock market volatilities.

1 Introduction

Text mining techniques have been used to perform event-driven stock prediction (Ding et al., 2015). The main idea is to learn distributed representations of structured events (i.e. event embeddings) from text, and use them as the basis to generate textual features for predicting price movements in stock markets. Here the definition of *events* follows the open information extraction literature (Fader et al., 2011; Yates et al., 2007), which has seen applications in semantic parsing (Berant et al., 2013), information retrieval (Sun et al., 2015) and text mining (Ding et al., 2014). Formally, an event is defined as a tuple (A, P, O), where A represents the agent, P represents the predicate and O represents the object. For example, "Microsoft profit rises 11 percent" can be represented as the event tuple (A ="Microsoft profit", P ="rises", O ="11 percent"). In addition, the main advantages of *event embeddings* include (1) they can capture both the syntactic and the semantic information among events and (2) they can be used to alleviate the sparsity of discrete events compared with one-hot feature vectors. The learning principle is that events are syntactically or semantically similar should have similar vectors.

The *event embedding* method of Ding et al. (2015) is based on *word embeddings* of the agent, predicate and object of an event. Neural tensor networks are used to combine the embeddings of the three components into embedding vectors of events. For training, one component of gold-standard event is randomly flipped to synthesize negative examples. This form of event embedding method suffers from some limitations. First, the obtained event embeddings cannot capture the relationship between two syntactically or semantically similar events if they do not have similar word vectors. On the other hand, two events with similar word embeddings, such as "Steve Jobs quits Apple" and "John leaves Starbucks" may have similar embeddings despite that they are quite unrelated. One important reason for the problem is the lack of background knowledge in training event embeddings. In particular, if it is known that "Steve Jobs" is the CEO of "Apple", and "John" is likely to be a customer at "Starbucks", the two events can have very different embeddings according to their semantic differences.

*This work was done while the first author was visiting Singapore University of Technology and Design

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2133–2142, Osaka, Japan, December 11-17 2016.

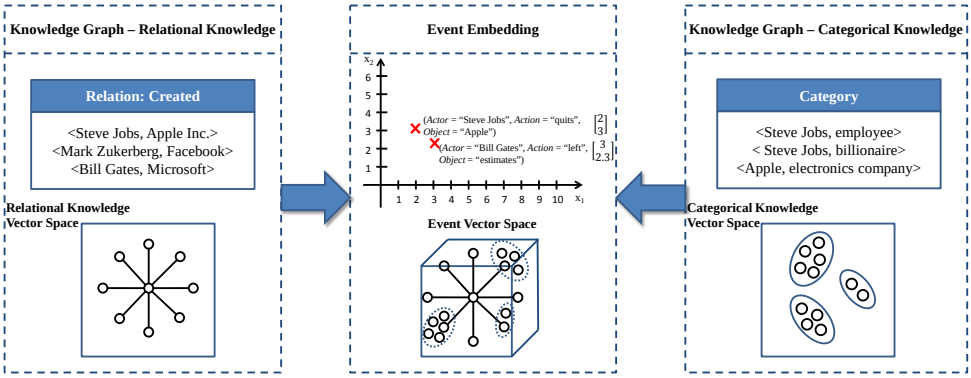


Figure 1: Incorporating knowledge graph into the learning process for event embeddings.

We propose to incorporate the external information from knowledge graphs, such as Freebase (Bollacker et al., 2008) and YAGO (Suchanek et al., 2007), into the learning process to generate better event representations. A knowledge graph stores complex structured and unstructured knowledge, and usually contains a set of vertices representing *entities* and a set of edges corresponding to the *relations* between entities. It commonly contains two forms of knowledge: *categorical knowledge* and *relational knowledge*. While categorical knowledge encodes the attributes and properties of certain entities, such as "programmer", "employee" for the person Mark Zuckerberg, relational knowledge encodes the relationship between entities, such as hasEconomicGrowth, isCEOof, etc. Both categorical knowledge and relational knowledge are useful for improving event embeddings. More specifically, categorical knowledge can be used for correlating entities with similar attributes, and relational knowledge can be used to differentiate event pairs with similar word embeddings.

We propose a novel framework for leveraging both categorical knowledge and relational knowledge in knowledge graphs for better event representations. As shown in Figure 1, we propose a coherent model to jointly embed knowledge graph and events into the same vector space, which consists of three components: the events model, the knowledge model, and the joint model. A neural tensor network is used to learn baseline event embeddings, and we define a corresponding loss function to incorporate knowledge graph information, by following recent work on multi-relation models (Socher et al., 2013).

Large-scale experiments on a YAGO corpus show that incorporating knowledge graph brings promising improvements to event embeddings. With better embeddings, we achieve better performance on stock prediction compared to the state-of-the-art methods.

2 Related Work

Stock Market Prediction There has been a line of work predicting stock markets using text information from daily news (Lavrenko et al., 2000; Schumaker and Chen, 2009; Xie et al., 2013; Peng and Jiang, 2015; Li et al., 2016). Pioneering work extracts different types of textual features from news documents, such as bags-of-words, noun phrases, named entities and structured events. Ding et al. (2014) show that structured events from open information extraction (Yates et al., 2007; Fader et al., 2011) can achieve better performance compared to conventional features, as they can capture structured relations. However, one disadvantage of structured representations of events is that they lead to increased sparsity, which potentially limits the predictive power. Ding et al. (2015) propose to address this issue by representing structured events using event embeddings, which are dense vectors. This paper proposes to leverage ground truth from knowledge graph to enhance event embeddings.

Knowledge Graph Embedding Recently, several methods have been explored to represent and encode knowledge graph (Bordes et al., 2013; Bordes et al., 2014; Chang et al., 2013; Ji et al., 2015; Lin et al., 2015) in distributed vectors. In this line of work, each entity is represented as a d-dimensional vector and each relation between two entities is modeled by using a matrix or a tensor. Most existing methods learn knowledge embeddings by minimizing a global loss function over all the entities and relations in

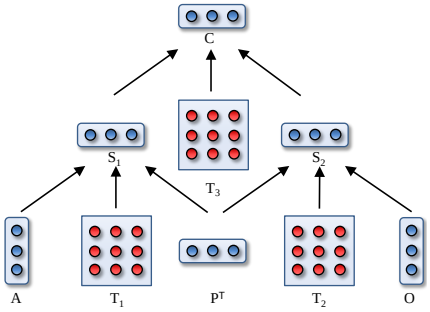


Figure 2: Baseline event-embedding model.

a knowledge graph. Entity vectors can encode global information over the knowledge graph, and hence are useful for knowledge graph completion (Socher et al., 2013). In this paper, we encode entity vectors into the learning process for event embeddings, so that information of knowledge graphs can be used for event-driven text mining and other tasks. Socher et al. (2013) has shown that previous work (Bordes et al., 2011; Jenatton et al., 2012; Bordes et al., 2012; Sutskever et al., 2009; Collobert and Weston, 2008) are special cases of their model, which is based on a neural tensor network. We follow Socher et al. (2013) and use tensors to represent relations in knowledge graph embeddings.

Our work is also related to prior research on joint embedding of words and knowledge graphs (Xu et al., 2014; Wang et al., 2014; Tian et al., 2016; Yang et al., 2014). Such work focuses on injecting semantic knowledge into distributed word representations, thus enhancing their information content. The resulting embeddings of words and phrases have been shown useful for improving NLP tasks, such as question answering and topic prediction. In comparison, our work integrates knowledge into vector representations of events, which was shown more useful than words for certain text mining tasks.

3 Knowledge-Driven Event Representations

We begin by introducing the baseline event embedding learning model, which serves as the basis of proposed framework. Then, we show how to model knowledge graph information. Subsequently, we describe the proposed joint model by integrating knowledge into the original objective function to help learn high-quality event representations. At the end of this section, we introduce the training process of the proposed framework in details.

3.1 Event Embedding

The goal of event embedding is to learn low-dimension dense vector representations for event tuples $E = (A, P, O)$, where P is the action or predicate, A is the actor or subject and O is the object on which the action is performed. We take the neural tensor network model of Ding et al. (2015) as the basis of our proposed framework. The architecture of neural tensor network for learning event embeddings is shown in Figure 2, where the bilinear tensors are used to explicitly model the relationship between the actor and the action, and that between the object and the action.

The inputs of the neural tensor network (NTN) are the word embeddings of A, P and O, and the outputs are event embeddings. We learn an initial word representation of d-dimensions ($d = 100$) from a large-scale financial news corpus, using the skip-gram algorithm (Mikolov et al., 2013). As most event arguments consist of several words, we represent the actor, action and object as the average of their word embeddings, respectively, allowing the sharing of statistical strength between the words describing each component (e.g. *Nokia's mobile phone business* and *Nokia*).

From Figure 2, $S_1 \in \mathbb{R}^d$ is computed by:

$$S_1 = g(A, P) = f\left(A^T T_1^{[1:k]} P + W \begin{bmatrix} A \\ P \end{bmatrix} + b\right), \quad (1)$$

where $T_1^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ is a tensor, which is a set of k matrices, each with $d \times d$ dimensions. The bilinear tensor product $A^T T_1^{[1:k]} P$ is a vector $r \in \mathbb{R}^k$, where each entry is computed by one slice of the tensor ($r_i = A^T T_1^{[i]} P, i = 1, \cdots, k$). The other parameters are a standard feed-forward neural network, where $W \in \mathbb{R}^{k \times 2d}$ is the weight matrix, $b \in \mathbb{R}^k$ is the bias vector, and $f = tanh$ is the activation function. S_2 and C in Figure 2 are computed in the same way as S_1.

We also experiment with randomly initialized word vectors as the input of NTN, which are commonly used in previous work on structured embeddings from knowledge graphs (Bordes et al., 2011; Jenatton et al., 2012). In our case, pre-trained word embeddings give slightly better results as compared with randomly initialized embeddings.

We assume that event tuples in the training data should be scored higher than corrupted tuples, in which one of the event arguments is replaced with a random argument. Formally, the corrupted event tuple is $E^r = (A^r, P, O)$, which is derived by replacing each word in A with a random word w^r in our dictionary $\mathcal{D}$ (which contains all the words in the training data) to obtain a corrupted counterpart A^r. We calculate the *margin loss* of the two event tuples as:

$$L_{\mathcal{E}} = loss(E, E^r) = \max(0, 1 - g(E) + g(E^r)) + \lambda \|\Phi\|_2^2, \tag{2}$$

where $\Phi = (T_1, T_2, T_3, W, b)$ is the set of model parameters. The standard L_2 regularization is used, for which the weight λ is set as 0.0001. The algorithm goes over the training set for multiple iterations. For each training instance, if the loss $loss(E, E^r) = \max(0, 1 - g(E) + g(E^r))$ is equal to zero, the online training algorithm continues to process the next event tuple. Otherwise, the parameters are updated to minimize the loss using standard back-propagation (BP) (Rumelhart et al., 1985).

3.2 Knowledge Graph Embedding

Knowledge graph embedding is mainly used to encode whether two entities (e_1, e_2) are in a certain relationship R (e.g. (Steve Jobs, Apple Inc.) has a relation "created"). To incorporate categorical knowledge, we expand the definition of (e_1, R, e_2) so that e_2 can also be an attribute of e_1 and R can also be an attribute type (e.g. (Steve Jobs, Profession, CEO)). Inspired by recent studies on learning distributed representations of multi-relational data from knowledge graph (Socher et al., 2013), we use a neural tensor network framework for knowledge graph embedding.

There are two significant differences when a neural tensor network is used for knowledge graph embedding, compared to when it is used for event embedding. First, we model a relation type in a knowledge graph by using a tensor rather than a vector. This is because the number of relation types in knowledge graph is limited, and using a tensor can increase the expressive power. Second, we use a simpler neural tensor network model to learn knowledge graph embedding, which is easier to train. In contrast, the baseline event embedding model uses a recursive neural tensor network architecture to preserve the original structure of events.

The neural tensor network replaces a standard linear neural network layer with a bilinear tensor layer, which directly relates the two entity vectors across multiple dimensions. The model computes the probability that two entities are in a certain relationship by the following function:

$$g(e_1, R, e_2) = \mu_R^T f\left(e_1^T H_R^{[1:k]} e_2 + V_R \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + b_R\right), \tag{3}$$

where $f = tanh$ is the activation function, applied element-wise, $H_R^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ is a tensor that consists of a set of $d \times d$ matrices, and the bilinear tensor product $e_1^T H_R^{[1:k]} e_2$ results in a vector $x \in \mathbb{R}^k$, where each entry is computed by on slice $i = 1, \cdots, k$ of the tensor: $x_i = e_1^T H_R^{[i]} e_2$. The other parameters for the relation R are the standard form of a neural network, where $V_R \in \mathbb{R}^{k \times 2d}$ and $b_R \in \mathbb{R}^k$.

The main method for training relation embeddings in knowledge graphs is similar to that for training the baseline event embeddings — each relation tuple in the training set $T^{(i)} = (e_1^{(i)}, R^{(i)}, e_2^{(i)})$ should receive a higher score than a corrupted tuple, in which one of the entities is replaced with a random

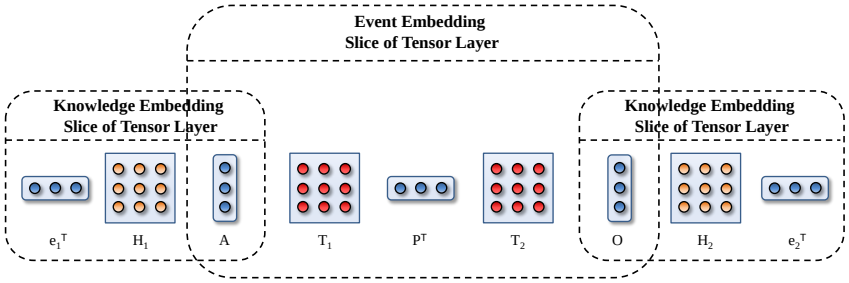


Figure 3: Architecture of the joint embedding model (only showing the tensor layer).

entity. Given a gold-standard tuple $T^{(i)} = (e_1^{(i)}, R, e_2^{(i)})$, the corresponding corrupted tuple is denoted as $T_c^{(i)} = (e_1^{(i)}, R^{(i)}, e_c^{(i)})$, where $e_c^{(i)}$ is the corrupted version of $e_2^{(i)}$. The set of all parameters is $\Omega = \{\mu, H, V\}$. We minimizing the following objective:

$$L_{\mathcal{K}} = \sum_{i=i}^{N} \sum_{m=1}^{M} max\left(0, 1 - g(T^{(i)}) + g(T_c^{(i)})\right) + \lambda \|\Omega\|_2^2, \tag{4}$$

where N is the number of training tuples, and the training objective scores the correct relation tuple higher than its corrupted version up to a margin of 1. For each correct tuple, we sample M randomly corrupted counterparts. We use standard L_2 regularization of all parameters, weighted by the hyperparameter λ.

3.3 Joint Knowledge and Event Embedding

Given a training event corpus $\mathcal{E}$ and a set $\mathcal{K}$ of relation tuples extracted from a knowledge graph, our model jointly minimizes a linear combination of the loss functions on both events and knowledge:

$$L = \alpha L_{\mathcal{E}} + (1 - \alpha) L_{\mathcal{K}} \tag{5}$$

where $\alpha \in [0, 1]$ is a model parameter to weight the two loss functions (the best development results were obtained with $\alpha = 0.4$). $\mathcal{E}$ and $\mathcal{K}$ share the same parameters — the embedding vectors for entities in events and their corresponding entities in knowledge graph are required to be the same.

Figure 3 shows the architecture of the proposed joint model. The algorithm is centralized on the event tuple (A, P, O), which is used to train the tensor values T_1, T_2 and T_3 in Figure 2. Two relations in the knowledge graph, H_1, H_2, which share entities A and O with the event, are shown on the two sides of Figure 3, respectively. By the shared entities A and O between events and knowledge relations. T_1, T_2, T_3, H_1 and H_2 can be trained simultaneously. Here relational knowledge can help incorporate information of the learned event representations by utilizing ground-truth relations between entities. Categorical knowledge can encode the entity information of the learned event representations by clustering similar entities together. The base event embedding model can preserve the structures of events.

Training The joint model is trained by taking the derivatives of the joint objective function with respect to the four groups of parameters T_1, T_2, H_1 and H_2, respectively. We have four derivatives for the i'th slice of the full tensor:

$$\frac{\partial g(e_1, R, A)}{\partial H_1^{[i]}} = \mu_i f'(z_i) e_1 A^T, z_i = e_1^T H^{[i]} A + V_i \begin{bmatrix} e_1 \\ A \end{bmatrix} + b_i; \frac{\partial g(A, P)}{\partial T_1^{[i]}} = \mu_i f'(z_i) A P^T, z_i = A^T T^{[i]} P + W_i \begin{bmatrix} A \\ P \end{bmatrix} + b_i$$

$$\frac{\partial g(O, R, e_2)}{\partial H_2^{[i]}} = \mu_i f'(z_i) O e_2^T, z_i = O^T H^{[i]} e_2 + V_i \begin{bmatrix} O \\ e_2 \end{bmatrix} + b_i; \frac{\partial g(P, O)}{\partial T_2^{[i]}} = \mu_i f'(z_i) P O^T, z_i = P^T T^{[i]} O + W_i \begin{bmatrix} P \\ O \end{bmatrix} + b_i$$

Table 1: Statistics of datasets.

	Training	Development	Test
#documents	442,933	110,733	110,733
#words	333,287,477	83,247,132	83,321,869
#events	295,791	34,868	35,603
time interval	02/10/2006 - 18/06/2012	19/06/2012 - 21/02/2013	22/02/2013 - 21/11/2013

z_i denotes the i'th element of the hidden tensor layer. We use minibatched L-BFGS (Nocedal, 1980) for optimization, which converges to a local optimum of our non-convex objective function. The rest of the model parameters, including μ, T_3, W, b and V, are trained in the same way as the baseline single models, for which the derivatives are calculated using standard back-propagation. It is the shared entities A and O that allow information exchange in training the values of T_1, T_2, H_1 and H_2, thereby improving the vector representations of structured events through relational and categorical knowledge.

4 Experiments

The performance of our knowledge-powered event embedding model is compared with state-of-the-art baselines by evaluating the quality of learned event embeddings on two tasks: event similarity and stock market prediction.

4.1 Experimental Settings

We use publicly available financial news from Reuters and Bloomberg over the period from October 2006 to November 2013, released by Ding et al. (2014). There are 106,521 documents in total from Reuters News, from which we extracted 83,468 structured events. From Bloomberg News, there are 447,145 documents, from which 282,794 structured events are extracted.

The structured events are extracted from news text using Open IE (Fader et al., 2011) and dependency parsing (Zhang and Clark, 2011), by strictly following the method of Ding et al. (2015). The timestamps of the news are also extracted, for alignment with stock price information. We conduct stock market prediction experiments on predicting the Standard & Poor's 500 stock (S&P 500) index and its individual stocks, obtaining indices and prices from Yahoo Finance. Detail statistics of the training, development (tuning) and test sets are shown in Table 1. For training knowledge-driven event embeddings, we use YAGO as the knowledge graph. The full knowledge graph consists of 10 million entities and 120 million facts, in more than 100 predefined relation types. We extract a sub knowledge graph of two thousand entities and more than 30 thousand relations for the experiments, which contains knowledge relevant to our news event data.

4.2 Task Description

Event Similarity We investigate whether the similarity between vector event representations is consistent with human-labeled event similarity, and whether better event representation is more useful for stock prediction. As there is no publicly available event similarity evaluation data, we conduct a set of human evaluations. Each event pair is associated with three independent human judgments on similarity and relatedness on a scale from 0 to 5, where 0 means that the event pair is completely dissimilar, and 5 indicates that the event pair has a strong similarity relation. For example, (Steve Jobs, quits, Apple) and (Steve Ballmer, quits, Microsoft) received an average score of 4.6, while (Steve Jobs, quits, Apple) and (John, leaves, Starbucks) received an average score of 0.4.

Similarity scores are computed by cosine similarity of embedding vectors for each event pair, based on which a ranked list is constructed. It is compared to the ranked list produced by the manual similarity scores according to human judgments. To evaluate the consistency between two ranking lists, we use Spearman's Rank Correlation ($\rho \in [-1, 1]$). A higher ρ corresponds to better event representation vectors. We compare our knowledge-driven event embedding (denoted as KGEB) with the baseline method proposed by Ding et al. (2015) (denoted as EB), and discrete event vectors with semantic lexicons based generalization method proposed by Ding et al. (2014) (denoted as DE).

Table 2: Experimental results on event similarity and its effect on S&P 500 index prediction. The improvement is significant at $p < 0.05$.

Methods	Spearman's Rank Correlation	Acc	MCC
DE	0.437	58.83%	0.1623
EB	0.591	64.21%	0.4035
KGEB	**0.616**	**66.93%**	**0.5072**

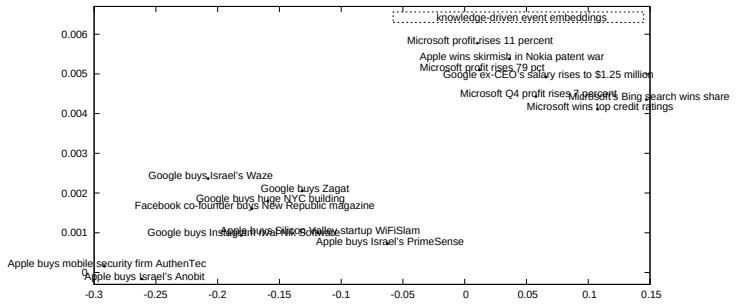


Figure 4: Two-dimensional PCA projection of 100-dimensional knowledge-driven event vectors.

Stock Prediction The stock prediction task can be treated as a binary classification problem. Following Tetlock et al. (2008), we automatically align 1,782 instances of daily trading data with news documents from the previous day. Specifically, we use the news information in day $t-1$ to predict price movements of stock market in day t. The output classification result [Class +1] represents that the stock closing price will increase compared with the opening price in day t, and [Class -1] represents that the stock closing price will decrease compared with the opening price in day t. Following Das and Chen (2007) and Xie et al. (2013), the standard measure of accuracy (Acc) and Matthews Correlation Cofficient (MCC) are used to evaluate the performances on S&P 500 index prediction and individual stock prediction.

The baseline methods are three state-of-the-art news-based stock market prediction systems: Luss and d'Aspremont et al. (2012) propose using bags-of-words to represent news documents, and constructing the prediction model by using Support Vector Machines (SVMs); Ding et al. (2014) report a system that uses structured event tuples $E = (A, P, O)$ to represent news documents, and investigate the complex hidden relationships between events and stock price movements by using a standard feedforward neural network; Ding et al. (2015) learn event embeddings for representing news documents, and build a prediction model based on a deep convolutional neural network. Ding et al. (2015) show that deep convolutional neural networks (CNN) are more powerful than SVMs and standard feedforward neural networks. As a result, we use CNN as the prediction model.

4.3 Results

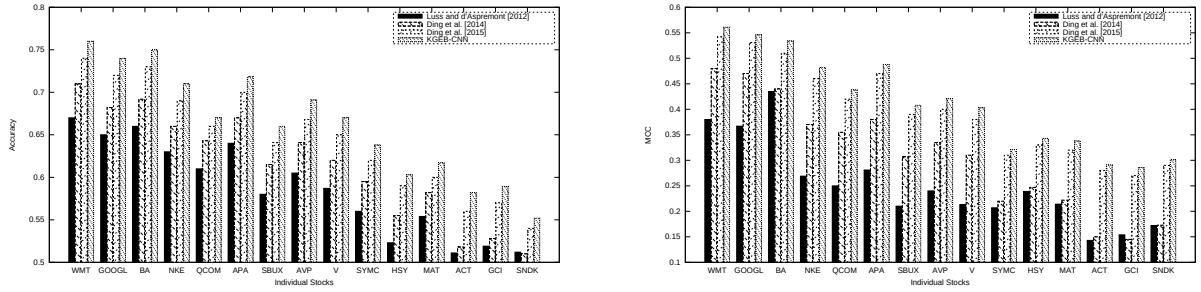
Event Similarity As shown in Table 2, we compare the performance of different event representation methods and their effects on S&P 500 index prediction. We find that although discrete event vectors are generalized by semantic lexicons (WordNet and VerbNet), the performance of KGEB is dramatically better than DE. This is mainly because the word coverage of semantic lexicons is limited, and the discrete representation is highly sparse. KGEB achieves better performance compared with EB on this task. The main performance gain results from better similarity between semantically related but lexically different events, thanks to the integration of knowledge. For example, "Steve Jobs quits Apple" and "John leaves Starbucks" have dissimilar vectors although they share similar word embeddings, as Steve Jobs is the CEO of Apple Inc. but John has no relationship with Starbucks encoded in knowledge graph. With the best event representation method, KGEB-based stock prediction achieves the best performance.

Figure 4 shows case studies on events about Google, Apple, Microsoft and Facebook. In particular, we apply two-dimensional PCA projection on the 100-dimensional event embeddings. It can be seen from the figure that by incorporating knowledge graph information, the joint model allows those events that correspond to the same semantic or topic to be close to each other.

S&P 500 Index Prediction We test the influence of knowledge-driven event embeddings on stock prediction by comparing KGEB with bag-of-words representations (Luss and d'Aspremont, 2012), structured

Table 3: Experimental results on index prediction.

	Acc	MCC
Luss and d'Aspremont (2012)	56.38%	0.0711
Ding et al. (2014)	58.83%	0.1623
WB-CNN	60.57%	0.1986
Ding et al. (2015)	64.21%	0.4035
KGEB-CNN	**66.93%**	**0.5072**



(a) Accuarcy (b) MCC

Figure 5: Experimental results on individual stock prediction (companies are named by ticker symbols).

event representations (Ding et al., 2014), baseline event embedding representations (Ding et al., 2015) and word embedding representations on the test dataset. A word embedding input (WB) consists of the sum of each word vector in a document; it addresses sparsity in word-based inputs, and can serve as a baseline embedding method. The experimental results are shown in Table 3. We find that:

(1) Comparison between the word-based models and event-based models (e.g. Luss and d'Aspremont (2012) vs Ding et al. (2014), WB-CNN vs Ding et al. (2015), WB-CNN vs KGEB-CNN) shows that events are more capable for representing news documents for stock prediction.

(2) Comparison between Ding et al. (2015) and KGEB-CNN shows that knowledge-driven event embeddings are more powerful than the baseline event embeddings. The main reason is that knowledge graph provides valuable ground-truth knowledge, which is helpful for learning better event embeddings. For example, "Chrysler recalls 919,545 Jeep SUVs" and "GM recalls nearly 474,000 cars" can be related, as "Chrysler" and "GM" are two automobile manufacturers, and Jeep SUV is a car model, which are recorded in the knowledge graph.

Individual Stock Prediction We compare our knowledge-driven event embeddings with the baseline methods on individual stock prediction, using the 15 companies selected by Ding et al. (2015) from S&P 500. The list consists of samples from high-ranking, middle-ranking, and low-ranking companies from S&P 500 according to the Fortune Magazine. The results are shown in Figure 5 (as space is limited, we only show comparison between KGEB-CNN and the three baselines). We find that knowledge-driven event embeddings achieve consistently better performances compared to the three baseline methods, on both S&P 500 index prediction and individual stock prediction. In most previous work, the accuracies of individual stock prediction are higher when only company-related news are used as inputs, compared with when sector-related news are used (Ding et al., 2014). This is because it is difficult to investigate the relationship among companies, and therefore news about other companies can be noise for predicting the stock prices of a company. However, knowledge graph can provide attributes of entities and relations between them, hence it is possible to learn more information from related companies to help decide the direction of individual stock price movements. For example, given that the news "GM recalls nearly 474,000 cars" leads to its stock price decrease in the training data, it can be predicted that Ford shares will fall next day according to the news "Ford is recalling about 433,000 2015 Focus" in the test data.

5 Conclusion

High-quality event representations are valuable for many text mining and NLP downstream applications. This paper proposed to incorporate knowledge graph into the learning process of event embeddings, which can encode valuable background knowledge. Experimental results on event similarity and stock

prediction showed that knowledge-powered event embeddings can improve the quality of event representations and benefit the downstream application.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, and gratefully acknowledge the support of the National Key Basic Research Program of China (973 Program) of China via Grant 2014CB340503, the National Natural Science Foundation of China (NSFC) via Grant 61472107 and 71532004, the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, October.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, pages 127–135.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *EMNLP*, pages 1602–1612.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Sanjiv R Das and Mike Y Chen. 2007. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management Science*, 53(9):1375–1388.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2014. Using structured events to predict stock price movement: An empirical investigation. In *Proc. of EMNLP*, pages 1415–1425, October.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Proceedings of IJCAI*, BueNos Aires, Argentina, August.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proc. of EMNLP*, pages 1535–1545.

Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. 2012. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems*, pages 3167–3175.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of ACL*, pages 687–696.

Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. 2000. Mining of concurrent text and time series. In *KDD-2000 Workshop on Text Mining*, pages 37–44.

Qing Li, Jun Wang, Feng Wang, Ping Li, Ling Liu, and Yuanzhu Chen. 2016. The role of social sentiment in stock markets: a view from joint effects of multiple information sources. *Multimedia Tools and Applications*, pages 1–31.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Ronny Luss and Alexandre d'Aspremont. 2012. Predicting abnormal returns from news using text classification. *Quantitative Finance*, (ahead-of-print):1–14.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jorge Nocedal. 1980. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782.

Yangtuo Peng and Hui Jiang. 2015. Leverage financial news to predict stock price movements using word embeddings and deep neural networks. *arXiv preprint arXiv:1506.07220*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, DTIC Document.

Robert P Schumaker and Hsinchun Chen. 2009. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):12.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Rui Sun, Yue Zhang, Meishan Zhang, and Donghong Ji. 2015. Event-driven headline generation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 462–472, July.

Ilya Sutskever, Joshua B Tenenbaum, and Ruslan R Salakhutdinov. 2009. Modelling relational data using bayesian clustered tensor factorization. In *Advances in neural information processing systems*, pages 1821–1828.

Paul C Tetlock, Maytal Saar-Tsechansky, and Sofus Macskassy. 2008. More than words: Quantifying language to measure firms' fundamentals. *The Journal of Finance*, 63(3):1437–1467.

Fei Tian, Bin Gao, En-Hong Chen, and Tie-Yan Liu. 2016. Learning better word embedding by asymmetric low-rank projection of knowledge graph. *Journal of Computer Science and Technology*, 31(3):624–634.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119. Citeseer.

Boyi Xie, Rebecca J. Passonneau, Leon Wu, and Germán G. Creamer. 2013. Semantic frames to predict stock price movement. In *Proc. of ACL (Volume 1: Long Papers)*, pages 873–883, August.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1219–1228. ACM.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 645–650.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. Textrunner: open information extraction on the web. In *Proc. of NAACL: Demonstrations*, pages 25–26. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.