DeepClue: Visual Interpretation of Text-based Deep Stock Prediction

Lei Shi, *Senior Member, IEEE,* Zhiyang Teng, Le Wang, Yue Zhang, and Alexander Binder

Abstract—The recent advance of deep learning has enabled trading algorithms to predict stock price movements more accurately. Unfortunately, there is a significant gap in the real-world deployment of this breakthrough. For example, professional traders in their long-term careers have accumulated numerous trading rules, the myth of which they can understand quite well. On the other hand, deep learning models have been hardly interpretable. This paper presents DeepClue, a system built to bridge text-based deep learning models and end users through visually interpreting the key factors learned in the stock price prediction model. We make three contributions in DeepClue. First, by designing the deep neural network architecture for interpretation and applying an algorithm to extract relevant predictive factors, we provide a useful case on what can be interpreted out of the prediction model for end users. Second, by exploring hierarchies over the extracted factors and displaying these factors in an interactive, hierarchical visualization interface, we shed light on how to effectively communicate the interpreted model to end users. Specially, the interpretation separates the predictables from the unpredictables for stock prediction through the use of intercept model parameters and a risk visualization design. Third, we evaluate the integrated visualization system through two case studies in predicting the stock price with financial news and company-related tweets from social media. Quantitative experiments comparing the proposed neural network architecture with state-of-the-art models and the human baseline are conducted and reported. Feedbacks from an informal user study with domain experts are summarized and discussed in details. The study results demonstrate the effectiveness of DeepClue in helping to complete stock market investment and analysis tasks.

Index Terms—Deep learning, visualization, model interpretation, stock prediction.

1 Introduction

Deep learning techniques [1] are reshaping the landscape of predictive analysis in the big data research area and have made major breakthroughs in image and speech recognition [2], question answering [3], machine translation [4] and many other application domains. In this paper, we focus on the financial analytics domain. It has been shown that texts such as financial news and tweets on stock markets are useful in predicting stock price movements. For example, financial news such as "Amazon profit beats forecasts" was accompanied with a surge of Amazon's stock price, while "Oil price hits a record high" triggered worries on the auto industry and weakened their performance in the stock market. Previous work has demonstrated an over 60% accuracy in predicting the daily stock price movement using deep neural networks over a large collection of financial news [5] [6] [7].

Nevertheless, end users can hardly benefit from these successful deep learning models in their primitive form. We consider two classes of users in this work: stock *traders* from public/private funds (or independent investors) who manage the stock trading operations; and stock market *analysts*, who provide the stock prediction models for traders. First, the everyday job of traders is to make trading decisions, i.e., to buy/sell which stock at which particular time. Such a decision is typically based on multiple sources of information known as trading signals, coming out of a large number of trading rules accumulated in the long term. To cope with the trader's job, there should be a method to help traders detect signals from the prediction model, so that traders can combine these signals with their traditional source of information to finalize the decision. The automatic stock trading based only on the prediction model can be an option, but it will require a much higher accuracy than that of the latest model. In some cases, a close to 60% accuracy can even lead to losses (Section 6.2). On the other hand, analysts' work is to fine-tune the stock price prediction model for particular stocks and market trends, in order to optimize the prediction accuracy. This will require analysts to have a deep understanding of failure cases of the prediction model.

To this end, both classes of end users will benefit from deep learning technology only if they can interpret the prediction model on where, when and why it works or does not work. This knowledge can then be assembled with the domain expertise to improve the investment in the stock market. Unfortunately, on interpretability deep learning models suffer from a well-known drawback in contrast to traditional machine learning methods such as linear regression and support vector machines (SVM). In some areas such as image recognition, the mechanism of deep learning has been partially known, e.g., working as level-of-detail feature selectors, from the basic visual feature up to motifs and finally to objects [8] [9]. For most other domains, there is still little clue on how deep learning models work. In our scenario, the use of text input introduces an additional word embedding stage to map text collections onto the feature space, which makes it more difficult to interpret the prediction model.

In this paper, we target the research problem of how to interpret text-based deep stock prediction model for end users, so that they can make up their stock trading decisions as well as improve the prediction model based on the interpretation. In particular, we investigate research questions including what kind of information can be efficiently extracted from the prediction model as interpretations, and how to communicate such information in an effective way to end users. Throughout this work, we depend

- *Lei Shi and Le Wang are with SKLCS, Institute of Software, Chinese Academy of Sciences, and UCAS. Email: {shil,wangle}@ios.ac.cn.*
- *Zhiyang Teng, Yue Zhang, and Alexander Binder are with Singapore University of Technology & Design. Email: zhiyang_teng@mymail.sutd.edu.sg, {yue_zhang,alexander_binder}@sutd.edu.sg.*

on an interactive visualization interface to bridge the prediction model and end users, which turns out a natural and straightforward choice. Yet, designing and prototyping such a visualization system can be quite challenging. First, traditional patterns discovered from data can be presented by visually distinct channels in the same data view, while in this case, the information extracted from the model lies in a higher order than the data pattern. Multiple coordinated views should be designed elaborately to illustrate the relationship among data, model, and interpretation. Second, the deep learning model is designed in a bottom-up structure to take advantage of the machine's capability in processing huge amount of data, while the visual information-seeking mantra is "overview first, details on demand" [10]. Third, it is commonly accepted that the stock market is information efficient [11], but not all stock price movements are predictable or reflected in text information. Ingenuity is required to separate predictable and unpredictable price changes.

In the literature, there is a recent surge on the topic of visualizing deep neural networks (DNN) for model interpretation. A large portion of these methods focused on the display of neural network architecture to help users understand the functionality of individual neurons and features [8] [12], interpret the mechanism of both small-scale neural networks [13] and large-scale multi-layer DNNs [14]. Another thread of research proposed to visualize the model output (e.g. the image class model [15]) or their correspondence to the input data through algorithms similar to back propagation [9]. While our study aligns with these successful methods on DNN model interpretation, the goal is fundamentally different. Instead of visually illustrating DNN structures, we target at extracting useful information from the prediction model, and incorporating this interpretation with domain expertise to improve the performance of stock trading and modeling. In addition, existing literature mostly studied model interpretation for image recognition and object detection tasks, while to our knowledge, we are the first to visually interpret the hidden linkage between public text collections and stock prices through deep learning models.

In summary, we make the following contributions.

- Based on a customized DNN architecture for stock price prediction (Section 3), we apply a model interpretation algorithm, i.e., the pixel-based layer-wise relevance propagation [16], to extract the textual factors relevant to the daily prediction result (Section 4.1). Notably, the extracted factors (i.e., keywords, bigrams, titles) are analyzed to form a factor hierarchy for effective visual interpretation by end users (Section 4.2).
- An integrated visualization system called DeepClue is designed and applied to the stock price prediction scenario, which visually correlates algorithm-extracted textual factors with stock price movements and the risks associated with the text-based prediction. Flexibilities are granted to end users in model configuration, factor analysis, and detailed reasoning. (Section 5)
- We evaluate the proposed system through real-life cases in analyzing the text-based deep stock prediction model built from financial news (Section 6.1) and social media collections (Section 6.2) on US stock markets. Quantitative experiments are conducted to compare the proposed neural network architecture with state-of-the-art models and the human baseline (Section 6.3). Informal user studies are then carried out with private-fund stock traders and deep learning model builders, which demonstrate the value of DeepClue in optimizing stock trading operations and improving the prediction model of stock price movement (Section 6.4).

2 Related Work

2.1 DNN Interpretation and Visualization

Early research related to the DNN interpretation can be found in Erhan et al. [8], who introduced the concept of understanding a particular unit of DNN by visualizing inputs that maximize the unit's response. This activation maximization method was compared with other alternatives including sampling the unit and linear combination of previous filters. Experiment results on image data sets showed that the activation maximization methods produced more interesting interpretation results. Zeiler et al. [9] proposed to map feature activations in neural nets back to inputs by deconvolution layers. Simonyan et al. [15] developed a class model visualization that generates a representative image for each class of interest, and a class saliency map for a single input image based on gradients with respect to the input pixel. Bach et al. [16] [17] introduced a class of algorithms named layer-wise relevance propagation (LRP), which decompose a neural net prediction layer by layer into scores for each neural unit, and applied it to state of the art deep networks in image classification. These scores, when computed for the inputs, explain the amount of contribution of a pixel or region to the prediction value for a given class. Dosovitskiy et al. [18] trained neural networks to reconstruct inputs from feature representations. Zintgraf et al. [19] developed an elaborate conditional sampling algorithm to analyze how deep neural networks respond to perturbed inputs. Yosinski et al. [12] introduced tools to visualize the activations on neural network layers, and the features extracted at each layer through regularization. Liu et al. proposed CNNVis [14], a visual analytics approach that employs layer and neuron clustering. CNNVis introduced several novel visualization algorithms such as hierarchical rectangle packing and matrix reordering to display features on clustered neurons.

Visualizations have made their way into deep learning toolboxes. Besides the well known deep dream [20], TensorFlow Playground by Google [13] provided an online visualization tool for non-experts to understand deep learning architecture and their training process through a direct manipulation design. Overall, previous literature on deep learning model visualization concentrates on the scenario of image classification with CNNs. DeepClue, in contrast to these systems, is dedicated to stock market investors for better understanding the association between text streams and stock price time series. Moreover, rather than opening the black-box structure of neural networks and interpreting the functionality of each individual unit, our method focuses on extracting input-level interpretable information from the DNN model and visually incorporating such information with domain expertise to improve the performance of stock trading and modeling.

2.2 Text-based Stock Prediction and Visualization

It has been pointed out by Kearney and Liu [21] that the complex and time-varying relationship between textual information and stock price poses an important area of study for financial analysis. The textual data for stock price prediction comes primarily from three data sources: public corporate filings, news articles, and the emerging social media content. This work focuses on the latter two sources. On using news articles for the prediction, Engelberg [22] and Tetlock et al. [23] employed firm-specific news from multiple

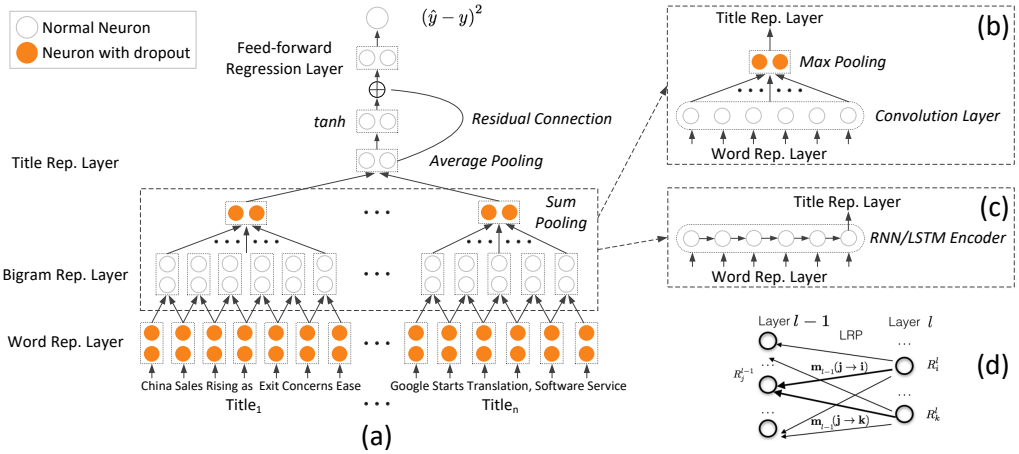


Fig. 1: Neural network architecture in this work: (a) the overall hierarchical structure featuring multiple representation layers from single word to bigram and news/tweet title; (b) an alternative convolution layer that can replace the bigram representation and the sum pooling on each title; (c) another design by a recurrent neural network layer with LSTM cells to represent each title as time series; (d) the mechanism of LRP algorithm that propogates relevance scores back to the input features over the neural network.

sources to predict firms' fundamentals, which inherently influence their stock prices. In addition, social media content, especially textual sentiments, has shown implicit effects on the stock market. The work by Chen et al. revealed that the views expressed on a popular social media site for investors have strong associations with the related firm's stock returns, thus helps to predict their stock price changes [24].

On stock market visualization, most state-of-the-art literature focused on the display of stock price time series. Similarity and cluster analysis have been employed to group the stock price time series into trajectories to optimize the visualization. For example, Ziegler et al. visually analyzed the distribution of time series trajectories among different market sectors [25]. Keim et al. presented the Growth Matrix visualization [26] for the simultaneous display of growth rates of all possible subintervals in a time series. Beyond the time series visualization, many other designs incorporate the related news and events to the display of stock price time series. Contextifier [27] produced annotated stock price visualizations given news articles as the information source. Sorenson and Brath proposed a system to visualize a large collection of stock-related events in a single view [28]. The event display can be visually correlated with the stock price time series for reasoning. Compared to the DeepClue visualization, most above works feature a direct visualization of raw stock price time series and the temporally correlated news/events. There has been little research on visualizing the predictive linkage between the stock price and the textual information, which is extracted from a state-of-the-art deep learning model.

3 Text-based Stock Price Prediction

3.1 Data Collection

We consider S&P 500 stocks in the US stock market from 2006 to 2015. Their historical prices are acquired from Yahoo Finance. We crawled financial news from Reuters and Bloomberg, obtaining in total 341,310 news articles. For each news, we extracted the title, textual content, and timestamp from their raw HTML file. To map each news to the corresponding stocks, we maintained a list of keywords for each firm (e.g., Apple: AAPL, AAPL.O, APPLE, AAPL.N, Apple Inc, etc.).

The stock-related tweets were collected through Twitter API in a period from April 2015 to November 2015, by matching the firm's cashtags in the tweet content. Cashtag [29] is a new way of sharing financial information on social media developed by Twitter and other providers. The firm's stock ticker symbols are prefixed with a dollar sign to compose the cashtag, e.g., Apple=$AAPL, Google=$GOOG. In total, we obtained 6,869,771 stock-related tweets. For each tweet, we recorded the create time, textual content, source, user, location and related firms.

3.2 Deep Neural Network Architecture

We take news data as an example to introduce the architecture of the neural network model adopted in this work. The model is built for each particular S&P 500 firm. The goal of the model is to predict a stock price $\hat{y}$ that is close to the real stock price y of the firm. The raw input of each model is the set of financial news titles collected on the target firm. Intuitively, news content can be useful for further enhancing the prediction accuracy. However, preliminary experiments using both news title and content as inputs (Supplemental Material–Table I) show that our model does not benefit from the additional content information, compared with only using the news title (Figure 11(a)). This is consistent with the observations of Ref. [5], who extract event information from both news title and content, showing that it does not substantially improve a model with only new title as the information source. Therefore, we leave it to future work to further exploit the usefulness of news content information.

Figure 1(a) shows our proposed deep regression model organized in a hierarchical neural network structure. The network consists of four layers: a word representation layer, a bigram representation layer, a title representation layer, and a feed-forward regression layer. The word representation layer accepts all the news titles as input and turns each word in the title into a real-valued word embedding vector [30]. The bigram representation layer constructs representation vectors for word bigrams based on the representation vector of individual words. The title representation layer summarizes representations of word bigrams and encodes each title into a title vector. The feed-forward regression layer receives the output of the title encoder and maps the output to a real-valued prediction through a feed-forward neural network with residual connections [31].

In addition to the prediction, the proposed model is also optimized for the interpretation purpose by three key designs.

First, we explicitly extract representation vectors (i.e., features) from the input news titles in a hierarchical, interpretable way (*word* $\rightarrow$ *bigram* $\rightarrow$ *title*), which provides the opportunity to efficiently visualize a large amount of contributing factors. Second, we make use of a combination of techniques to prevent overfitting, e.g. the dropout mechanism. Third, as the hierarchical method lengthens the backward propagation path, we introduce residual connections to ease the burden of training a deep neural network.

Note that the proposed deep stock prediction model can be upgraded by introducing state-of-the-art deep neural network structure, such as Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). In Section 6.3, we will describe two such alternative designs by replacing the bigram-based title representation method with convolution layers (Figure 1(b)) and RNN with LSTM cells (Figure 1(c)). The prediction performance of these alternative designs is also studied in Section 6.3.

3.2.1 Word Representation Layer

In the word representation layer, we adopt the distributed method proposed by Mikolov et al. [30], which embeds each word into the high-dimensional space as a real-valued dense vector. If two words' representation vectors are close to each other in the high-dimensional space, their corresponding words will have similar semantic meanings. For example, the distance between "rise" and "boost" is much closer than that between "rise" and "erase". Given a vocabulary V, the representation vector of the ith word is denoted by $\mathbf{v}_i$ with a length of d.

In the news data set, the vocabulary size is approximately 44K. Out-of-vocabulary words are replaced with a uniform token "UNK". The numerical tokens are replaced with the token "NUM". All the word representation vectors are pre-trained on the combined Bloomberg and Reuters news data set ($\sim$ 300M words). The word vector length is set to 50 (i.e., $d = 50$). To prevent overfitting, we apply the dropout mechanism [32] on the embedding process. The main idea of dropout is to stochastically deactivate a portion of neurons with a probability of p_{drop} to create an exponential ensemble of the neural network structure, in order to improve the generality of the neural network after training.

3.2.2 Bigram Representation Layer

Over the word representation layer, we capture the information of phrases using bigram, which is the sequence of two adjacent words in a sentence, e.g., "hit record", "draw investors". Consider a bigram B, in which the representation vector of its two words are $\mathbf{v}_i$ and $\mathbf{v}_j$. The representation vector of B, denoted by $\mathbf{u}_B$, is computed by

$$\mathbf{u}_B = \tanh(\mathbf{v}_i + \mathbf{v}_j) \tag{1}$$

With this design, the bigram representation captures both the semantic of phrases and their composing words, effectively going beyond the original bag-of-words model. As the summation is vector sum, the local feature on each word is well preserved. In comparison, the bigram vector computed by the distributed representation can be inaccurate in our case due to the sparsity of bigram phrases. The distributed bigram representation could also lose the semantic of individual words. Explicitly synthesizing bigrams and words together will introduce additional parameters, which causes overfitting.

The hyperbolic tangent function (tanh) is a standard activation function used in neural network units, which provides nonlinearity for the model. Compared with the sigmoid function (σ), the hyperbolic tangent function prevents the training from getting stuck when the input can be strongly negative, which is the case after the word embedding.

3.2.3 Title Representation Layer

The title representation layer maps each news title into a vector and then aggregates all the vectors on the same day into a single representation to summarize the overall input data of that day.

In the first step, consider a trading day when there are n news input. Denote the representation of the jth news title by the vector $\mathbf{t}_j$, the representations of all its K_j bigrams as $\mathbf{u}_{kj}$ ($1 \le k \le K_j$). Each title representation is constructed by a sum pooling layer as shown in Figure 1(a), and is mathematically computed by

$$\mathbf{t}_j = \sum_{1 \le k \le K_j} \mathbf{u}_{kj} \tag{2}$$

The sum pooling can effectively represent all local features of a sentence to predict stock price movements globally. Meanwhile, most financial news titles have a similarly short length by words (an average of 8.8 words and a standard deviation of 1.7 words in our data set). This pattern prevents the sum pooling from the significant bias toward overlong news titles.

In the second step, all the title vectors $\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n\}$ on the same day are summarized into a single vector $\mathbf{s}$ using an averaging pooling layer.

$$\mathbf{s} = \sum_{1 \le j \le n} \mathbf{t}_j / n$$

Compared with the concatenation method, the average pooling keeps the number of model parameters reasonably sized to reduce overfitting. On the other hand, compared with the sum pooling method, the average pooling is insensitive to the number of news in a day, which can vary a lot according to the trading cycle and the data collection mechanism. Before the average pooling layer, we also apply a block dropout operation [33] on each title with a probability p_{drop} to alleviate overfitting.

3.2.4 Feed-Forward Regression Layers

Finally, we design a regression layer with residual connection [31] to fit the representation of each day's news title (denoted as $\mathbf{s}$) into the next day's stock price change $\hat{y}$.

$$\begin{aligned} \mathbf{h} &= \tanh(\mathbf{W}^h \times \mathbf{s} + \mathbf{b}^h) + \mathbf{s} \\ \hat{y} &= \mathbf{w}^o \times \mathbf{h} + b^o \end{aligned} \tag{3}$$

Here $\mathbf{W}^h, \mathbf{b}^h, \mathbf{w}^o, b^o$ are parameters of the regression model. The hyperbolic tangent function (tanh) is used as the activation function because the linear transformation of $\mathbf{s}$ can also be negatively valued. A shortcut connection directly adds $\mathbf{s}$ to the output of the regression layer, which is effective for accelerating the gradient flow from the output vector $\mathbf{h}$ to $\mathbf{s}$ in the title representation layer.

For the loss function, we use the minimum squared distance plus a ℓ_2-regularizer. With N training days, let y_k be the ground truth price of the k-th day, the loss function is given by

$$L = \sum_{1 \le k \le N} (\hat{y}_k - y_k)^2 + \frac{\lambda}{2} ||\Theta||_2 \tag{4}$$

where λ is the weight decay parameter, $\Theta = \{\mathbf{M}, \mathbf{W}^h, \mathbf{b}^h, \mathbf{w}^o, b^o\}$ are model parameters.

TABLE 1: The training time of DeepClue prediction model (Apple Inc.'s financial news data set).

#Days	#Titles	#Words	Training Time (s)
1312	8122	70331	96.33
656	2043	16800	55.33
328	663	4854	44.33
164	311	2253	24.67

3.3 Model Training and Complexity

DeepClue is implemented with DyNet v1.0, a neural network software library optimized for natural language processing tasks [34]. The neural network model is trained using stochastic gradient descent with momentum [35]. The learning rate η_0 is set to 0.01 initially and we apply the method in [36] for the decay of learning rate, with a decay parameter of 0.1. The dimensionality of the *tanh* hidden layers is set to 50, same with the word vector length after embedding. All the model parameters are randomly initialized with a uniform distribution $[-\sqrt{\frac{6}{r+c}}, \sqrt{\frac{6}{r+c}}]$, where r and c are the number of input and output neurons respectively [37]. The dropout rate p_{drop} is set to 0.5 and the weight decay parameter λ is set to 10^{-4}. All these parameters are chosen from a combination of candidate options through experiments.

Theoretically, the training time complexity of the tree-like neural network model in Figure 1 is $O((p+d)\cdot I)$ where p is the number of parameters in the model, d is the input data size determined by the number of words, bigrams, titles and training days in the data set, I is the number of backpropagation iterations until convergence. Note that the word embedding is pre-trained, so its cost is not included. In DeepClue, p increases linearly with the number of training days and I is hard to model, therefore we can estimate the training time complexity as $O(d)$, linear to the input data size. Empirically, we have shown the training time on Apple Inc.'s financial news data set in Table 1, which approximately follows this linearity. Note that DyNet v1.0 does not support GPU, we expect the training process to be 7.1 times faster after we port the implementation to DyNet v2.0 and train the model with a single GPU. This speedup is estimated by running the basic MNIST routine on DyNet 2.0 with and without using GPU.

4 Model Interpretation

4.1 Relevant Keyword Extraction

We introduce a method to identify the importance of textual factors to the stock price change by analyzing the neural network model. The goal is to compute a **relevance score** with respect to the prediction result of each trading day, denoted as $f(\cdot)$, for each word, bigram, and news title. Take the word relevance score as an example, $f(\mathbf{v})$ defines how much contribution a word with the vector representation $\mathbf{v}$ has made to the stock price prediction. A positive (negative) score indicates that this word is an evidence for the rise (fall) of the stock price.

Many existing methods to extract such scores have been introduced in both the computer vision domain [9], [15], [16], [38], [39] and the natural language processing research community [40]. Existing methods can be classified into two main categories, namely gradient-based methods [8], [38] and relevance score based methods [16], [17], [40]. We choose the LRP algorithm because it has performed favourably in a quantitative evaluation by Samek et al. [41] and was shown to extract more plausible explanations in the NLP domain [40] compared to other gradient-based approaches.

The idea of LRP [16] is to propagate the relevance score from the final output layer to the input layer in a way similar to the back-propagation of gradients [42]. Take one trading day as an example when the predicted stock price change is denoted as f. LRP decomposes this change into the relevance score of all the neurons in each representation layer (e.g., words, bigrams, titles). Denote the ith neuron at the lth layer as R_i^l and its relevance score as f_i^l. At each layer l, these scores satisfy

$$f = \sum_i f_i^l \tag{5}$$

In each representation layer, the propagation rule is designed to distribute the relevance score according to the trained neural network parameters. Consider R_i^l, the ith neuron at the lth layer, its contribution to R_j^{l-1} at the layer $l-1$ is denoted by $\Delta f_{i\to j}^{l-1}$.

$$\Delta f_{i\to j}^{l-1} = \frac{m_{l-1}(j\to i)}{\sum_k m_{l-1}(k\to i)} f_i^l \tag{6}$$

where $m_{l-1}(j\to i)$ denotes the forward message passed from the neuron R_j^{l-1} to R_i^l in the trained neural network model. $\Delta f_{i\to j}^{l-1} \neq 0$ only if R_j^{l-1} and R_i^l are connected in the neural network.

Next, the overall relevance score of neuron R_j^{l-1} at the layer $l-1$ is summed by

$$f_j^{l-1} = \sum_i \Delta f_{i\to j}^{l-1} \tag{7}$$

Figure 1(d) shows the details of this propagation procedure in LRP.

In the regression layer, there are many feasible propagation rules. We adopt the $\varepsilon-$rule proposed by Bach et al. [16], due to its simplicity and excellent performance. For example, on the single layer network $h = \tanh(\mathbf{w}\times\mathbf{s}+b)$ where $\mathbf{s}$ is the input vector, $\mathbf{w}$ and b are neural network parameters, the relevance score $f(h)$ at the output layer is decomposed into the relevance score at all the neurons in the input layer, denoted as $f(s_k)$ for the kth neuron.

$$f(s_k) = f(h)\cdot\frac{w_k s_k}{(\mathbf{w}\times\mathbf{s}+b)+\varepsilon\cdot\mathrm{sgn}(\mathbf{w}\times\mathbf{s}+b)} \tag{8}$$

where ε is a small positive value for numerical stability when the value of $\mathbf{w}\times\mathbf{s}+b$ is close to zero. In this paper, ε is set to 10^{-4}.

Note that $\sum_k f(s_k) < f(h)$ because the intercept parameter b also accounts for the predicted stock price change. The contribution of b in the prediction can be seen as the momentum of stock prices determined by the non-text information. In another perspective, the relevance score assigned to b indicates the risk of using text information to predict stock prices. The relevance score $R(b)$ of the intercept term b is given by

$$f(b) = f(h)\cdot\frac{b}{(\mathbf{w}\times\mathbf{s}+b)+\varepsilon\cdot\mathrm{sgn}(\mathbf{w}\times\mathbf{s}+b)} \tag{9}$$

In our design, the relevance score of two intercept vectors $\mathbf{b^h}$ and b^o are summed to represent the risk of text-based prediction.

Finally, the overall relevance score of each word and bigram is obtained by summing up their relevance scores propagated from all the titles at all vector dimensions. Note that each relevance score is computed once every day according to the daily prediction result.

4.2 Factor Analysis

By the LRP algorithm, a list of words having nonzero relevance score can be obtained on each day, which is defined as keywords here. These keywords, together with relevant bigrams and news/tweet titles, compose the potential influencing factors to the change of stock price. A straightforward method to visualize these

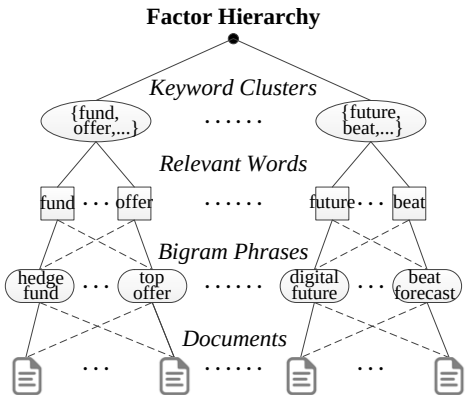


Fig. 2: The four-level factor hierarchy that expands the context of relevant keywords into keyword clusters, bigrams and documents.

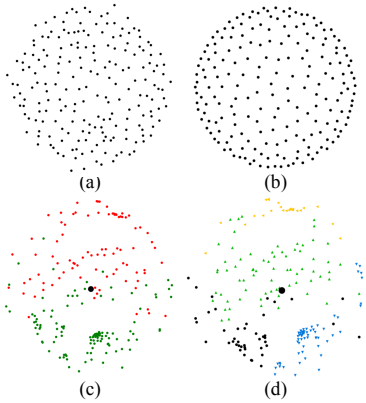


Fig. 3: MDS projection of 200 sampled keywords by: (a) keyword embedding vector; (b) keyword relevance time series; (c) nearest neighbor fused keyword relevance, the point color indicates the polarity of overall relevance; (d) the same projection of (c) after k-means clustering, color/shape indicates the cluster index.

factors is to juxtapose the top factors and their relevance score time series in a list view. This view can be aligned with the time series of the actual/predicted stock price changes for multi-factor analysis. Due to the large number of relevant factors (e.g., 1801 keywords for Apple/APPL), there is an obvious constraint that the list view can quickly grow beyond the limit of the screen space. These keywords can be shortlisted after sorting by the overall relevance (i.e., $\ell 1$ norm of the relevance vector), but an overview of all relevant factors will be missing.

To provide such an overview and allow users to drill-down to each interested factor, we propose to construct a factor hierarchy based on the list of relevant keywords extracted. As shown in Figure 2, the factor hierarchy is composed of four levels: the top level are keyword clusters that include all extracted relevant keywords; the second level are keywords themselves; the third level are bigram phrases stemmed from relevant keywords; and the bottom level are individual documents (news, tweets, etc.) containing these keywords/phrases. The lower two levels of the factor hierarchy can have overlaps in the same level. For example, one bigram phrase can yield two relevant keywords, and one document can have multiple bigram phrases. This factor hierarchy offers an initial overview of all factors relevant to stock price changes. The navigation on factor hierarchy through expand/collapse operations allows analyzing the details of every factor.

In constructing the factor hierarchy, a significant challenge is to appropriately cluster the keywords for an initial factor overview. There are two feasible keyword metrics for clustering: the keyword embedding vector that represents their semantics, and the relevance score time series indicating their contribution to the stock price change over time. We have investigated both metrics and projected their distribution into a 2D space using dimensionality reduction methods such as MDS [43] and tSNE [44]. With both the keyword embedding vector and the relevance score, the projection results is a uniform distribution on the unit ball (Figure 3(a)(b)). This is because the keyword distribution in both semantic and relevance score space is sparse. To address this issue, we introduce a fusing method to combine semantic and relevance information. For each keyword, we first find their n nearest neighbors in the semantic space, and then add the relevance score time series of this keyword and its n neighbors as the representation of the keyword. The fused projection result reveals more clustered structure. For example in Figure 3(c), the keywords with positive (negative) overall contribution stay close to each other according to the color mapping of the keywords. In this case by applying k-means clustering, the keywords are further grouped into four clusters, distributed in separate radial slices centric to the neutral keyword in the center, though overlappings are heavy. This result implies that each cluster can include the keywords that correlate in relevance score during a sub-interval of the selected time window. Here the number of clusters in k-means is determined by the elbow criterion [45].

The extracted keywords, clusters and bigrams can be compared in time series with the stock price movement. To serve this need, we compute the cross-correlation [46] between these time series.

5 Visualization

5.1 Design Principle

The DeepClue interface is composed of four coordinated views, as shown in Figure 4(a)(b)(c)(d). We follow two principles in the visualization design.

First, the visualization interface should help users complete three key tasks in the scenario of stock price prediction and analysis.

Understanding stock market: The baseline task is to examine the underlying stock data, including price movements, trading volume, historical rise&fall trends, and the potential temporal patterns.

Visualizing prediction result: Over the stock data, users should get access to the result produced by the model, i.e., whether a certain stock is predicted to rise or fall on the next day. S/he also needs to navigate the input data to the prediction model, i.e., the news/tweets collection in our scenario.

Interpreting prediction model: Finally, users are expected to unveil the myth of the model by learning why and how each rise&fall prediction is decided. In DeepClue, this is achieved by visualizing the key textual factors that jointly make up the decision.

Second, we design DeepClue for financial domain users, i.e., stock traders and investors. These users are mostly accustomed to classical financial visualization interfaces (e.g., Yahoo Finance), especially for the first two tasks in presenting stock data and their predictions. The classical visualization depends heavily on statistical charts. Therefore, to reduce the user's learning cost, we build DeepClue from commodity statistical charts, both in the stock data and prediction visualization (Figure 4(a)(c)) and in illustrating their relevant predictive factors (Figure 4(b)(d)).

5.2 Visualization Components

In details, the *stock timeline view* in Figure 4(a) displays the price movement of a selected stock over time. This view is organized

Fig. 4: DeepClue interface: (a) stock timeline view showing the stock price history in an overview+detail design; (b) factor hierarchy view displaying the relevant textual factors to the stock prediction in a hierarchical structure; (c) document list view showing the related documents with the selected factor; (d) keyword map view depicting the relationship of relevant keywords.

in an overview+detail design to support flexible navigation of the timeline. The overview timeline chart on the top row allows users to specify a focused window on the timeline. The selected timeline is enlarged in the detail chart on the bottom row of Figure 4(a). On the detail timeline chart, four time series can be shown in two tabbed groups, as indicated by the draggable legends in the top left corner of the detail timeline chart. The first tab on the left includes the *actual* stock price (solid line), the *predicted* stock price (dashed line), and the *risk* of the prediction (shading area). Note that there is little difference between these charts in Figure 4(a) because the selected timeline is almost ten years and the daily prediction introduces little variation. More separation of these time series can be observed in Figure 7 and Figure 10(a). In another legend tab on the right, the investment yield curve according to the prediction model is displayed, which is highly suggested in the expert study as the key indicator of model's success. As shown in Figure 9, the yield curve always starts from one. The model earns when the curve is above one and loses when it is below one.

Beneath the detail timeline charts in Figure 4(a), a series of bar charts seated on the time axis indicates the *actual* stock price movement, as indicated by the selected option button in the left control panel. The green (red) bars correspond to rise (fall) of stock prices. Using the same option button, the bar chart can be switched to display three other time series data: the predicted stock price movement, the prediction error, and the predicted stock price movement made by text information only.

Below the stock timeline view, the *factor hierarchy view* displays the relevant textual factors in the prediction of stock price, as shown in Figure 4(b). By vertically aligning the factor timeline with the stock timeline, the contribution of these factors to the stock price can be visually explained. The initial overview shows a list of top keyword clusters. The cluster can be expanded to a list of top keywords and then bigram phrases (Figure 6). Each keyword (cluster) and phrase is visualized by a time series of bar charts, which indicate the contribution of this factor to the stock prediction. Beneath the bar chart, a series of line bars represent the number of documents containing the factor. Another line chart is drawn over bar charts and line bars to display the cross-correlation score between the time series of the factor's contribution and the overall predicted stock movement.

When a time window is selected on the factor hierarchy view, the corresponding documents (news, tweets, etc.) are displayed in a *document list view* (Figure 4(c)). This view gives the basic information of all the relevant documents, including their title, date and individual contribution to the prediction. The document content can be unfolded upon a click of the document title.

The relationship of all relevant keywords are also revealed by a *keyword map view* in Figure 4(d). These keywords are displayed by a 2D projection of the keyword relevance time series. Each dot in this map corresponds to a keyword. The color hue of the dot indicates the positive (green) or negative (red) relevance to the stock prediction. Only the name of keywords with top positive and negative relevance are displayed to avoid the visual clutter. All the other keywords and their overall relevance can be shown as tooltip upon a mouse hovering on the dot.

5.3 User Interactions for Model Analysis

The basic user interactions in DeepClue are the customization of four coordinated views according to the analysis task. On the stock timeline view, the "+" on the left panel allows users to juxtapose two bar chart time series for comparison purpose (Figure 4(a)). These bar charts can be dragged vertically to avoid overlaps with the timeline chart. In the factor hierarchy view, the hierarchical structure can be expanded and collapsed. Each factor can be deleted or popped to the top. New factors can be added with the manual input on the left panel. All factors can be re-sorted according to different criteria, e.g., positive/negative relevance or the correlation score. In the document view, the list can be sorted by either the publication date or their contribution to the stock price change.

Beyond view customizations, another suit of interactions is designed to analyze the linkage between the stock price prediction and the impacting textual factors in multiple levels.

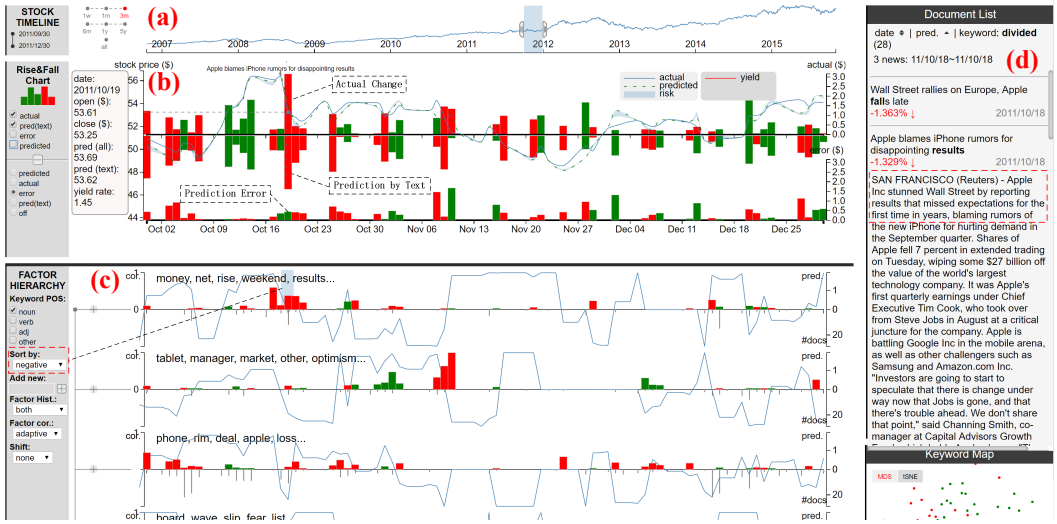


Fig. 5: The prediction of the fall of Apple's stock price on 10/19/11. The contributing factor of the weak financial result and the negative news on "Apple Inc. ... reporting results that missed expectations for the first time in years" are detected in DeepClue.

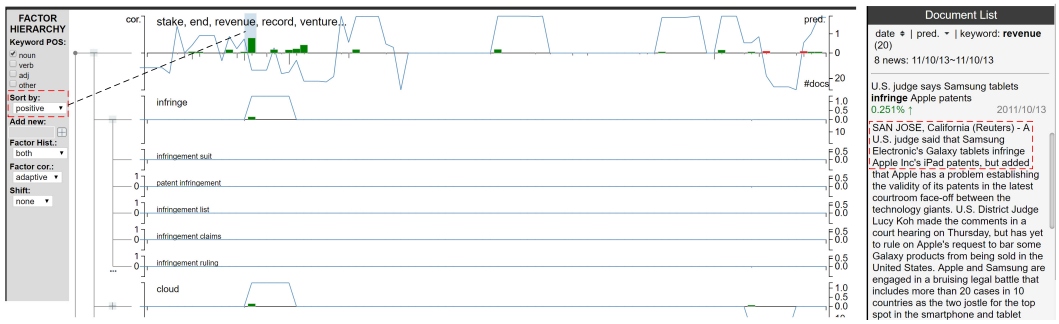


Fig. 6: The prediction of the rise of Apple's stock price on 10/14/11. The factor hierarchy and news relevant to patent issues are located.

Model Selection: On the top-right corner of the user interface (Figure 4), users can choose one stock index to analyze the deep learning model of a specific firm. Furthermore, s/he can switch between different textual data sources for the prediction model, by the option button box to the left of the stock index selector. The choices include the financial news (default), firm-related tweets crawled from Twitter, and the annual and quarterly financial reports. In the left panel of the factor view, we allow users to configure a part-of-speech filter to only show the relevant factors with the selected part-of-speeches, such as noun, verb, adjective, etc. In the keyword map view, different projection algorithms (MDS and tSNE) can be selected.

Timeline Selection: The system supports multiple ways to select a specific time window to study. In the stock timeline overview panel, a double-ended range selector is provided to specify the detailed time window shown in all the other panels. Users can zoom and pan with this timeline selector, or use shortcut buttons to specify a pre-defined time window (a week, a month, etc.). On the stock timeline detail panel and the factor hierarchy view, users can follow up to select an inner time window to display more detailed context, i.e., the documents containing the selected factors inside the inner time window.

Timeline Comparison: Comparing multiple time series is a key task for interpreting the relevant predictive factors. We have designed several interactions for this task. In the stock timeline view, the actual and predicted stock price changes can be juxtaposed vertically as two bar chart time series for contrasting. An explicit prediction error timeline can also be displayed. On the top-row bar charts, another stock time series data can be displayed as the reversed bar chart. Most often, this interaction helps to compare the predicted stock movement and its component made by textual information only (Figure 4(a)).

In the factor view, when an inner time window is selected, users can sort the factors by their contribution within this window and visually compare the top factors with the stock price timeline. When a keyword cluster is hovered on, the keyword map view is re-drawn to highlight the keywords belonging to the cluster.

6 Evaluation

6.1 Learning from Financial News

In the first case, we study the use of DeepClue in interpreting the stock price prediction model over financial news. The details of the data set and model can be found in Section 3. We invited a stock trader from a private fund, one of our target user, to work with the DeepClue system. He was interested in investing on Apple Inc. (NASDAQ: AAPL), so the DeepClue configuration is set to display the prediction of Apple's stock price from 2006 to 2015.

The initial DeepClue view provided to the trader is shown in Figure 4. At the lower part of the interface (Figure 4(b)), four most significant keyword clusters are listed, sorted from top to bottom according to their overall contribution to the prediction, in which only noun keywords are considered. The keyword cluster in the first row represents a group of positive keywords, indicated by the series of green bars, which have the largest contribution to the up prediction of the stock price. These keywords contain "court, street, record, ...", probably related to the patent issues of Apple Inc. and its wall street expectations. The next positive keyword cluster is in

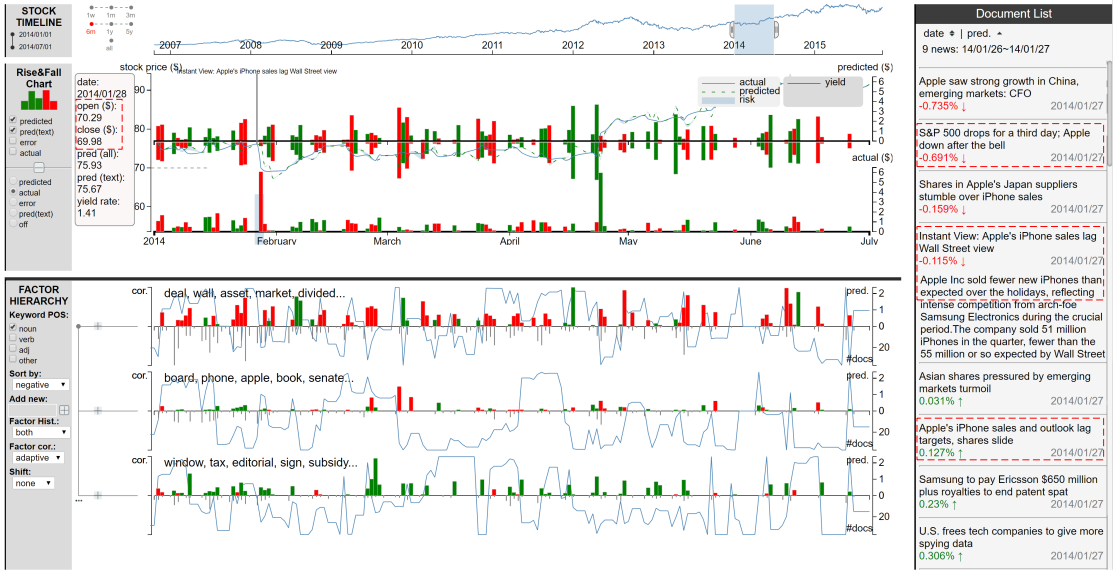


Fig. 7: The missed prediction of Apple's big price fall on 01/28/14. The reasons are revealed to be the ambiguous classification of similar news and the overlooked price movement in the after-hour trading.

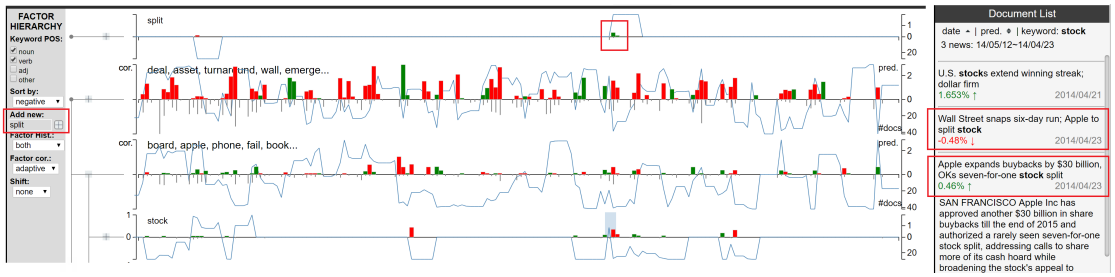


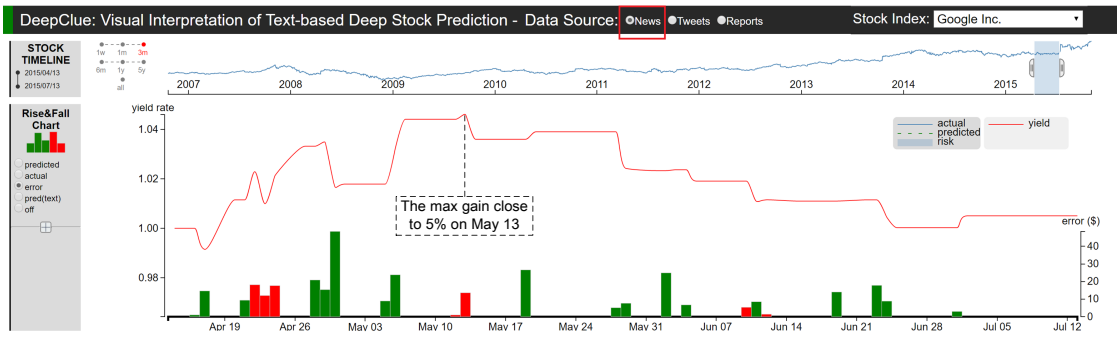
Fig. 8: The missed prediction of Apple's price inflation on 04/24/14. The reason is attributed to some news titles mixing multiple events.

the third row, composed of "report, china, profit, ...", which state the impact of the Chinese market and profit reports. The major negative keyword clusters are "fall, invest, share, ..." (second row) indicating that profit or sale is fallen and there are negative trends in Apple's investor or investment. The last keyword cluster by "surface, rival, ..." implies that the competitor's product such as Microsoft's surface can be negative to Apple's performance.

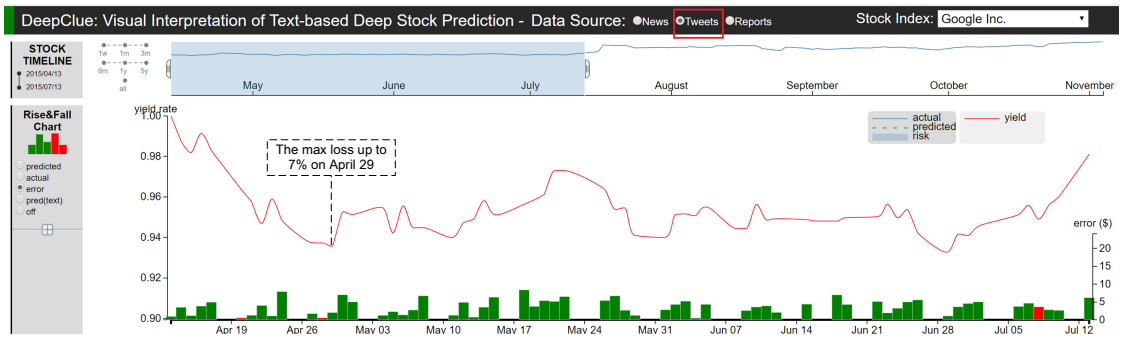
After an overview of all possible hints to the investment, the trader would like to drill-down to locate individual signals that can help him make a particular trading decision, i.e., when to buy/sell Apple's stock. Due to the nature that the stock price movement is predicted on a daily basis, he positioned himself on the short-term investment. Based on his domain knowledge, he first located the time periods when Apple's stock price varied quite a lot. Trading at these periods will have a better chance of success for the short-term investment. As shown in Figure 5(a), he chose the period from 09/30/11 to 12/30/11, immediately after Apple's co-founders Steve Jobs passed away. He put the focus on the days when the actual change is significant, the prediction error is small, and the risk of prediction is low, i.e., most of the predicted change is contributed by the textual data. These conditions can be examined through the rise&fall charts of the stock timeline view in Figure 5(b). The trader first picked 10/19, when Apple's price dropped $3.16 and it is largely predicted by textual factors. After sorting all keyword clusters by their negative contributions on 10/19, the trader identified a key cluster that leads to this fall. As shown in Figure 5(c), the related keywords are "money, results, ...". The corresponding document view in Figure 5(d) listed three financial news on 10/18, all contributing to the negative prediction. Notably, the second news (expanded) suggested the root cause - "Apple Inc. stunned Wall Street by reporting results that missed expectations for the first time in years". This key linkage was validated by the CNN market analysis in the next day: http://money.cnn.com/2011/10/19/markets/markets_newyork/. By analyzing with DeepClue, the trader consolidated the most important signal for trading Apple's stock price. Though being the most profitable company in the world, its stock price is still very sensitive, or even more, to the financial results, especially the unusual ones.

In a similar analysis, the trader studied 10/14, when there was a significant rise in Apple's stock price by $1.81 and the change was correctly predicted. To analyze the signals, the trader sorted all keyword clusters positively on the time window of 10/14. After examining the first keyword cluster and the related financial news effective on that day, as shown in Figure 6, the key signal was found to be the patent case. First, the list of keyword factors sorted positively in the left panel of Figure 6 reveals "infringe" and further phrases such as "infringement suit", etc. Second, the related news list on the right panel of Figure 6 includes "A U.S. judge said that Samsung Electronic's Galaxy tablets infringe Apple Inc's iPad patents." This signal indicated that, while Apple was involved in multiple patent issues with other companies in the same industry, the result on the court has a good impact on Apple's stock price.

On the financial news case, we also invited a stock analyst whose task was to improve the prediction model and obtain better accuracy measures. He moved on to diagnose the failure of



(a) By Financial News



(b) By Tweets

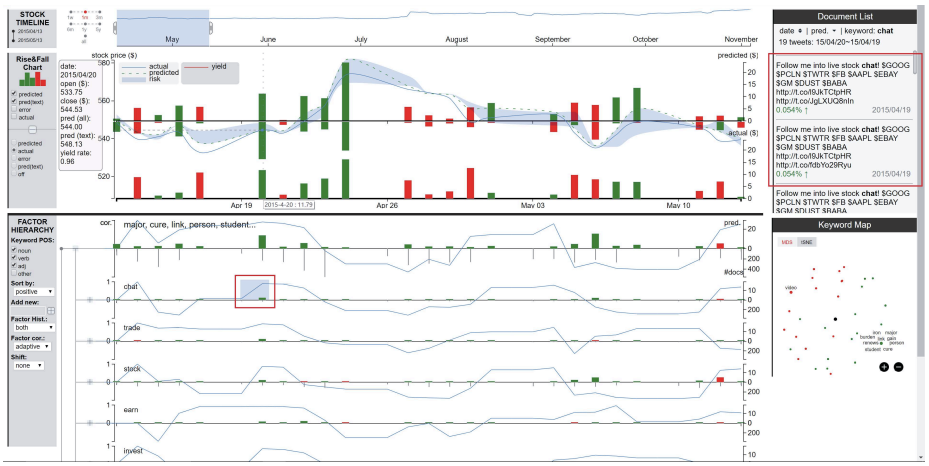
Fig. 9: The yield curve in deploying the prediction model to Google's stock price change from April to July, 2015. Two models using the financial news and Twitter messages related to Google are compared.

prediction models. On a selected 6-month period in the first half of 2014, he examined two large price deviations on 01/28/14 and 04/24/14, on both days the model incorrectly predicted the stock price changes (Figure 7, Figure 8). On 01/28, Apple's price fell 7% after it reported an iPhone sale far weaker than expected. The text-based predictor forecast a slight overall drop of $0.39, but could not envision this big loss. The news list contributed to the prediction in Figure 7 states part of the reason. While there are correct, negative signals like "Apple's iPhone sales lag Wall Street view", the similar news is misunderstood as positive: "Apple's iPhone sales and outlook lag targets". Interestingly, there is one news: "S&P 500 drops for a third day; Apple down after the bell", which indicates that the actual price change happens mostly in the after-hour trading of 01/27. In Figure 7, it can be observed that the actual drop during the trading day of 01/28 only reaches $0.32, close to what is predicted. The analyst then realized one noteworthy model-building deficiency: the after-hour trading is not considered in the training phase, i.e., only the changes within the daily trading hour are predicted while the financial news used can also affect the after-hour trading. In an optimized design, the prediction can be based on the stock price change between the closing time of adjacent trading days. In Supplemental Material–Table IV, we present the experiment results using different price movement definitions. The optimized design is shown to achieve better performance on average compared with the DeepClue model trained by the trading day price movement.

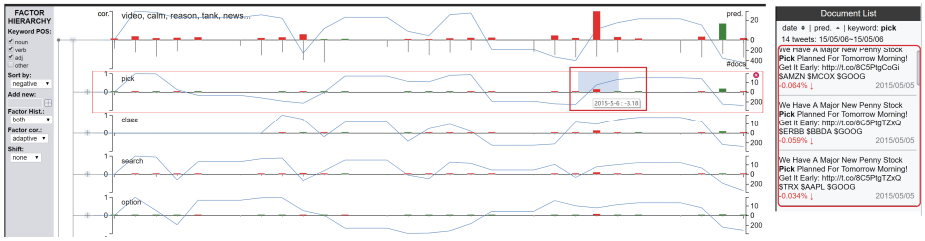
On the other day when Apple's price inflates (04/24), the prediction is on the contrary slightly negative. The analyst checked CNN reports on that day and found the root cause of inflation to be the announcement of the increase in both stock buybacks and dividends. Therefore, in analyzing the model factors, all verb keywords were included in the factor hierarchy in order to capture the stock-related action. After a few interactive analysis, the analyst concluded a key finding on the failure of the prediction model on that day. As illustrated in Figure 8, the events of stock buyback and split, are shown to contribute to the rise of the stock price, after adding the keyword of "split" to the factor view for analysis. However, some news mixing multiple events in their title have affected the positive prediction. As shown in the right panel of Figure 8, the news of "Wall Street snaps six-day run; Apple to split stock" was considered negative because, in the first half of the title, Wall Street stocks are said to stop to rise, though this overall trend in the market should not disturb the positive prediction on Apple's stock split event. After checking all the titles of Apple's news in the data set, there are 443 out of 12134 news having semicolons to compose a single title from multiple clauses. We manually cleaned these titles to only leave the clauses relevant to Apple's stock. In total, there are 319 titles modified (262/26/31 in the training/development/test data), a selected list is given in Supplemental Material–Table V. Training with the modified news data set, we obtain an increase of prediction accuracy on Apple's stock price movement (+3.7% on the training data, +1.2% on the development data, +0.7% on the test data), compared with the standard DeepClue model evaluated in Section 6.3. The development/test performance is less affected probably because there are fewer news titles modified due to the semicolon.

6.2 Learning from Social Media

In the second case study, we built the prediction model with the firm's Twitter message stream as input (Section 3.1). We are curious about which kind of data source, financial news or Twitter, are better for the deep learning model in terms of the investment return. With this motivation, we invited a stock analyst to work on the prediction model over Google/Alphabet Inc. (NASDAQ: GOOG) during the period from 04/13/15 to 07/13/15. To reason on the investment return, the system is configured to display the yield



(a) The rise on 04/20/15



(b) The drop on 05/06/15

Fig. 10: Interpreting the stock prediction model on Google. On both days above, the prediction is shown to be overfitted.

curve in the stock timeline view (Figure 9). If the yield curve moves above one, you are making money, otherwise, you are losing.

In the first trial, the analyst switched between the prediction model built from the two data sources, i.e., the financial news and the Twitter stream. Figure 9(a)(b) show that the two models have rather diversified yield curve, especially in the first month of the time period under study. The Twitter model lost up to 7% money (Figure 9(b)) while the news model gained as much as 5% in the same time (Figure 9(a)).

To reason on this difference, the analyst selected the first month (i.e., April 2015) to interpret the Twitter model with DeepClue. The initial view (Figure 10(a)) suggests that the risk of the Twitter-based prediction seems to be higher than that of the news prediction, as shown by the large shading area on the timeline chart. This is also indicated by the visual fact that the overall prediction accuracy (22/23) is much higher than the accuracy with textual factors only (15/23), a sign of overfitting. In more details, all keyword factors are grouped into two clusters. The analyst drilled down to study three days when there were significant rise/fall on the stock price (04/20, 05/06, 05/12). On 04/20, Google's stock price went up 2.2% ($11.79) but there was no clue about the root cause in WWW. The Twitter-based prediction was correct on that day (up $15.39) and most of the prediction came from the first cluster ($13.64). After the analyst expanded the first cluster and sorted the keywords by their positive contribution on that day, four keywords were shown to have the largest contribution ("chat":$2.48,"trade":$2.48,"stock":$2.38,"earn":$1.75). The analyst checked the detailed tweet list containing each keyword on that day. The interpretation was not convincing in that most tweets are unrelated to the stock price, e.g., most "chat" tweets were live stock chat ads, as shown in Figure 10(a). The same fine-grained study was conducted on 05/06 and 05/12 respectively. On both days, the top bigram was "stock pick", i.e., the spamming tweets for selling stocks (Figure 10(b)). Interestingly, the contribution of "stock pick" tweets was negative on one day ("penny stock pick") and positive on the other day (just "stock pick"). These findings suggested an overfitted prediction behavior. The analyst could then concluded that though the prediction accuracy by the Twitter model can be a little better than the news model, the Twitter model turned out to lose money because of its overfitting to a large amount of low-quality Twitter messages. More data cleansing needs to be carried out on the Twitter stream before building the stock prediction model.

6.3 Quantitative Experiment

In this part, we report on a set of quantitative experiments that evaluate the stock price prediction performance using the proposed neural network architecture, in comparison to alternative network designs. The baseline prediction accuracy by the human is measured in a controlled user study. Significant level is set at 0.05.

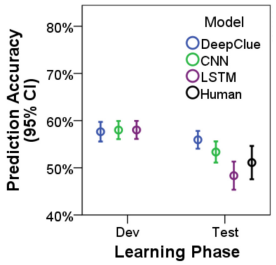
Neural network architecture and experiment setup. The neural network model of *DeepClue* is shown in Figure 1(a). The network is designed in a hierarchical, interpretable architecture that extracts the textual features (i.e., factors) in a bottom-up manner, from words, bigrams, to titles, and the title collection of a single trading day. The basic pooling functions are used to compute the title representation from the output of word embeddings.

Alternatively, the convolution operation [1] can be applied over the word representation to extract bigram and sentence-level features, as shown in Figure 1(b). These local features are processed through a max pooling layer to generate the representation of each title. The network structure further above follows that of DeepClue, and the entire architecture is called the *CNN*-based design.

Further, we also implement a third design for comparison, as shown in Figure 1(b). Each title representation is computed by

Model	Phase	Days	Accuracy (%)									
			AAPL	GOOG	BAC	WMT	GM	XOM	BA	GSPC	T	Average
DeepClue	train	1425	83.8	87.8	**93.8**	52.5	83.9	90.3	91.2	87.6	46.2	79.7
	dev	178	56.1	58.7	55.9	59.2	**62.8**	59.4	57.4	53.7	55.6	57.6
	test	177	56.4	58.1	52.4	55.1	56.8	52.4	**60**	56.4	55.8	55.9
LSTM	train	1425	80.8	67.3	74.2	88.8	**95.3**	87.7	76.3	54.9	73.5	77.6
	dev	178	56.1	57.3	57.3	58.7	**63**	56.1	60.3	58.6	54.8	58.0
	test	177	50.6	42	51.1	44.6	50.1	**52.8**	52	47.6	44.2	48.3
CNN	train	1425	90.8	50.6	50.1	49.4	**91.4**	46.5	51	54.8	51.3	59.5
	dev	178	53	58	55.6	60.3	58	57.9	59.8	58.1	**61.3**	58.0
	test	177	54.3	55.3	52	50.8	53.8	**58.5**	54	48	53.4	53.3

(a)



(b)

Fig. 11: The prediction accuracy over the stock price movement of nine companies in S&P 500. The news data is used. (a) The comparison of three alternative models in the training, development and test phases; (b) The corresponding chart of prediction accuracy distribution (95% CI), the result of human prediction is also presented as a baseline.

an LSTM encoder in the same way as in the natural language translations [47]. This is called the *LSTM*-based design.

In the performance experiment, the financial news data set of ten years (Section 3.1) are divided into the training set (1,425 days), development set (178 days), and test set (177 days). The stock price movement classification accuracy (ACC) is adopted as the performance metric: $ACC = \frac{\#\text{correct predictions}}{\#\text{total predictions}}$. When the polarity of the predicted price movement and the ground truth are the same, we count the current prediction as correct; otherwise as incorrect. Both the predicted price change and the ground truth (i.e., the actual price change) are defined as the daily change from the closing price of the previous day to the closing price of the present day.

Note that in our designs, the neural network can also take the news of multiple days as input and predict the stock price movement of a longer period than one day. According to the finding in [23] [48] [5], the predictive performance with only the previous day's information is better than the weekly and monthly prediction because of the efficient market hypothesis [11]. We conduct experiments to validate this finding and the result can be found in Supplemental Material–Table II&III. Finally, in this work, all the three alternative designs leverage the title information of one previous day to predict the stock price movement of the current day. This ensures a fair comparison of the neural network design.

User experiment on the human baseline. To obtain a baseline prediction performance, we conducted a controlled user experiment to understand how good financial users can achieve in the stock price prediction task given financial news. As it is difficult to obtain the performance of the best human working on our data set (e.g., professional traders in private funds), we recruited graduate students majoring in international finance to participate this study, who can be seen as the average user in the domain.

In total, we recruited 24 users in this experiment, organized into four separate groups. Each user is given $\sim$30 daily prediction tasks on Apple's stock price movement, and the six users in each group cover the 177-day test data set exactly once. For each user under a particular daily prediction task, all Apple's financial news titles in the previous day are provided in a web page and the user is required to select the prediction of "rise" or "drop" for the current day's stock price using a radio button. The result of all these selections is compared with the actual stock price movement to derive the prediction accuracy of each user. Before a user entered the formal experiment, we design a sample prediction task on the training data set to allow users to warm-up. The link to the user experiment document is available in Supplemental Material–Appendix B.

Result and analysis. The result is summarized in Figure 11(a), which lists the performance of three alternative models in predicting the stock price movement during the training, development and test phases. On each row of Figure 11(a), nine representative S&P 500 firms are predicted under the same setting, and each cell indicates the ACC of one predictive model over a particular firm. The highest ACC in each row is depicted in bold to highlight the top performance among different firms, i.e., 63% in the development phase (LSTM on GM) and 60% in the test phase (DeepClue on BA). Note that the training set accuracy is less useful in representing the model performance, so we do not discuss it in details. In Figure 11(b), the ACCs on each row are treated as a probability distribution and further depicted with error bars (95% confidence interval), grouped by models and learning phases. On the test set, we also depict the distribution of ACCs by human prediction obtained in the controlled user experiment.

In general, we have two findings on the experiment result. First, in all cases, DeepClue leads to comparable or even higher prediction accuracy compared with the other two models (i.e., CNN and LSTM). By the analysis of variance (ANOVA) test, there are a few cases that significant differences are observed among these models. In the test phase ($F(2,24) = 13.56, p < .001$), the ACCs of DeepClue (55.93$\pm$2.45%) and CNN (53.34$\pm$2.93%) are significantly higher than that of LSTM (48.33$\pm$3.89%) according to the Tukey post hoc test ($p < .001, p = .007$). No significant difference is observed in other pairwise comparisons.

Second, ANOVA test shows that compared with the human baseline (51.1$\pm$8.36%), the ACC of DeepClue (55.93$\pm$2.45%) is significantly higher, $F(1,30.2) = 6.52, p = .016$. Because of the non-compliance with the homogeneity of variances ($p < .05$ in the Levene test), we have applied the Welch ANOVA here.

Discussion. Overall, under the current stock price prediction scenario and the financial news data set, DeepClue performs the best among alternative models that have been implemented. CNN and LSTM do not necessarily perform as well as what we have expected. There can be certain deficiencies of these models. For example, CNN introduces additional parameters in the convolution layer that is hard to train with only $\sim$1000 training samples. LSTM that considers longer context than the bigram seems to be inappropriate in predicting the stock price movement, as the stock price is mostly affected by the local feature of the news title, e.g., key event or concept [6], but not the whole title.

We believe that the main reason for the failure of advanced models lies in data noise. First, there are many financial news in the data set which are irrelevant to the price change of the next day. For example, the background report of a firm can appear in any day and includes historical events of the firm that may affect its previous but not the current stock price. Second, there are a lot of cases that the root cause of the price change is not disclosed in the public reports. Therefore, one stock with all positive news in one day can

have its stock price drop in the next day. Fitting these noisy cases will potentially downgrade the prediction accuracy, in addition to the overfitting issue on a particular data set. The interactive model analysis by DeepClue is a potential way to address this data noise issue. In the studies of Section 6.1 and Section 6.2, we have detected several cases that the training data is irrelevant or even misleading with respect to the stock price movement. Third, the delayed release of news, irrationality, and insider trading can also cause noise in the data set. For conceptual simplicity in visualization, we choose the current model structure.

6.4 Expert Feedback

In this work, we also invited two groups of domain experts in multiple rounds of informal pilot studies. Each study is composed of a training session to instruct users on how to use the system, a test session when users complete several tasks in their target analysis scenario, and finally, a discussion session to collect their feedbacks on the system.

As the first user group, we visited a private fund company and invited three traders to use our system to assist in their daily tasks. In their everyday job, they depend heavily on machine learning models, though most are not as complex as the deep learning model (e.g., SVM). They frequently build multi-factor models [49], and therefore require to evaluate the performance of each factor, for the selection of factors to include in the model and the right stocks to invest. Time is extremely valuable for them, so a comprehensive visualization interface serves the central need of their work. After the pilot usage, traders acknowledged the main advantage of DeepClue to be the good adaptation to their analysis flow. Initially, an overview of all signals are provided, then a smaller time window can be selected, e.g., the period when the model fails. After that, the group of factors with the worst performance can be quickly identified, drilled-down and exported, completing a loop of analysis. They also provide valuable suggestions for the future development of DeepClue. First, instead of the stock price timeline, a yield curve can be drawn in the overview panel. Users can locate the time when the model earns/loses money. This yield curve has been implemented in the latest version of DeepClue. Second, in addition to the keyword embedding view, they demand some charted summary of all factors. Third, for the prediction model, more trading information can be taken in beyond the text message, e.g., trading volume and transactions. We consider the last two suggestions to be our future work. A detailed expert study report is attached in Supplemental Material–Appendix A.

On the other hand, we visited researchers who designed the text-based stock prediction algorithm. They are interested in evaluating their models with the interactive interface of DeepClue. The primary feedback they have after using the system is, beyond the standard evaluation by statistical performance measures (e.g., prediction accuracy), DeepClue allows to drill-down to the detailed factors and go back to the stock price timeline in history, to understand the root cause of daily prediction results. This has significant implications in detecting and understanding the overfitting effect. In more details, they obtain several useful findings for improving their prediction model. First, while each word is treated equally as the input feature, its POS class can have a large impact on the word's role in the prediction model. For example, the noun words can be interpreted as better relevant factors to the stock price movement than other classes of words such as verbs and adjectives. Second, the prediction with Twitter messages sometimes lead to a higher accuracy than financial news, but it is found to be mostly uninterpretable. We hypothesize the performance gain to come from the smaller number of trading days to predict and the larger number of daily messages in the Twitter data set, compared with the news data set. These factors lead to a more overfitted model.

7 Conclusion

We present DeepClue, a system that visually interprets text-based deep learning models in predicting stock price movements. DeepClue integrates three key designs from the cutting-edge deep learning technology: a hierarchical neural network model that embeds semantics in intermediate processing layers for interpretation; a backpropagation-like algorithm that effectively distributes the decision of prediction back to individual documents, bigrams and words; and an interactive visualization interface that allows users to navigate and analyze stock price timelines, textual factors, and their correlations. DeepClue has been deployed to predict S&P 500 stocks using mainstream financial news and firm-specific tweets. Both case studies, quantitative experiments, and the informal user study with domain experts demonstrate the usefulness of the proposed system in learning from, evaluating and improving the text-based deep stock prediction models.

Acknowledgments

Lei Shi and Le Wang are supported by China National 973 Project 2014CB340301, NSFC Grant 61772504, and the Key Research Program of Frontier Sciences, CAS (Grant No. QYZDY-SSW-JSC041). Alexander Binder expresses gratefulness for the support by the SUTD startup grant, and by the ST-SUTD corporate laboratory on cybersecurity. Yue Zhang is the corresponding author and is supported by NSFC Grant 61572245. We would like to thank Dan Zhang for helping to conduct the experiments in the supplemental material.

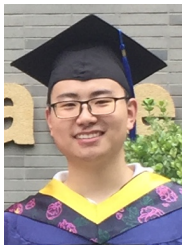
References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS'12*, 2012, pp. 1097–1105.
[3] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," in *EMNLP'14*, 2014, pp. 615–620.
[4] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR'15*, 2015.
[5] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Using structured events to predict stock price movement: An empirical investigation," in *EMNLP'14*, 2014, pp. 1415–1425.
[6] ——, "Deep learning for event-driven stock prediction," in *IJCAI'15*, 2015, pp. 2327–2333.
[7] Y. Peng and H. Jiang, "Leverage financial news to predict stock price movements using word embeddings and deep neural networks," in *NAACL'16*, 2016, pp. 374–379.
[8] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, 2009.
[9] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV'14*, 2014, pp. 818–833.
[10] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *IEEE Symposium on Visual Languages*, 1996, pp. 336–343.
[11] B. G. Malkiel and E. F. Fama, "Efficient capital markets: A review of theory and empirical work," *The journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.
[12] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv:1506.06579*, 2015.
[13] "Tensorflow playground," <http://playground.tensorflow.org/>.

[14] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu, "Towards better analysis of deep convolutional neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 91–100, 2017.
[15] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv:1312.6034*, 2013.
[16] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.
[17] A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller, and W. Samek, "Layer-wise relevance propagation for neural networks with local renormalization layers," *ICANN'16*, vol. 9887, pp. 1–9, 2016.
[18] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," *arXiv:1506.02753*, 2015.
[19] L. M. Zintgraf, T. S. Cohen, and M. Welling, "A new method to visualize deep neural networks," *arXiv:1603.02518*, 2016.
[20] "Google deep dream," <http://deepdreamgenerator.com/>.
[21] C. Kearney and S. Liu, "Textual sentiment in finance: A survey of methods and models," *International Review of Financial Analysis*, vol. 33, pp. 171–185, 2014.
[22] J. Engelberg, "Costly information processing: Evidence from earnings announcements," in *AFA San Francisco Meetings*, 2008.
[23] P. C. Tetlock, M. Saar-Tsechansky, and S. Macskassy, "More than words: Quantifying language to measure firms' fundamentals," *The Journal of Finance*, vol. 63, no. 3, pp. 1437–1467, 2008.
[24] H. Chen, P. De, Y. J. Hu, and B.-H. Hwang, "customers as advisors: the role of social media in financial markets," in *BFC'13*, 2013.
[25] H. Ziegler, M. Jenny, T. Gruse, and D. A. Keim, "Visual market sector analysis for financial time series data," in *VAST'10*, 2010, pp. 83–90.
[26] D. A. Keim, T. Nietzschmann, N. Schelwies, J. Schneidewind, T. Schreck, and H. Ziegler, "A spectral visualization system for analyzing financial time series data," in *EuroVis'06*, 2006, pp. 195–202.
[27] J. Hullman, N. Diakopoulos, and E. Adar, "Contextifier: Automatic generation of annotated stock visualizations," in *CHI'13*, 2013, pp. 2707–2716.
[28] E. Sorenson and R. Brath, "Financial visualization case study: Correlating financial timeseries and discrete events to support investment decisions," in *IV'13*, 2013, pp. 232–238.
[29] "Twitter cashtag," <http://money.cnn.com/2012/07/31/technology/twitter-cashtag/>.
[30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS'13*, 2013, pp. 3111–3119.
[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv:1512.03385*, 2015.
[32] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[33] W. Ammar, G. Mulcaire, M. Ballesteros, C. Dyer, and N. A. Smith, "One parser, many languages," *arXiv:1602.01595*, 2016.
[34] G. Neubig and Others, "DyNet: The dynamic neural network toolkit," *arXiv preprint arXiv:1701.03980*, 2017.
[35] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," *Journal of Machine Learning Research*, vol. 28, pp. 1139–1147, 2013.
[36] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, "Transition-based dependency parsing with stack long short-term memory," *arXiv:1505.08075*, 2015.
[37] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks." in *AISTATS'10*, vol. 9, 2010, pp. 249–256.
[38] J. Li, X. Chen, E. Hovy, and D. Jurafsky, "Visualizing and understanding neural models in nlp," in *NAACL'16*, 2016, pp. 681–691.
[39] G. Montavon, S. Bach, A. Binder, W. Samek, and K. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *CoRR*, vol. abs/1512.02479, 2015.
[40] L. Arras, F. Horn, G. Montavon, K.-R. Müller, and W. Samek, "Explaining predictions of non-linear classifiers in nlp," in *ACL Workshop on Representation Learning for NLP*, 2016, pp. 1–7.
[41] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller, "Evaluating the visualization of what a deep neural network has learned," *IEEE Transactions on Neural Networks and Learning Systems*, 2016.
[42] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[43] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.
[44] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.
[45] D. J. Ketchen and C. L. Shook, "The application of cluster analysis in strategic management research: an analysis and critique," *Strategic management journal*, vol. 17, no. 6, pp. 441–458, 1996.
[46] Wikipedia, "Cross-correlation," <https://en.wikipedia.org/wiki/Cross-correlation>.
[47] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
[48] B. Xie, R. J. Passonneau, L. Wu, and G. G. Creamer, "Semantic frames to predict stock price movement," in *ACL'13*, 2013, pp. 873–883.
[49] E. F. Fama and K. R. French, "Common risk factors in the returns on stocks and bonds," *Journal of financial economics*, vol. 33, no. 1, pp. 3–56, 1993.

Lei Shi is a professor in the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. Previously, he was a research staff member and research manager at IBM Research - China. He holds B.S. (2003), M.S. (2006) and Ph.D. (2008) degrees from the Department of Computer Science and Technology, Tsinghua University. His research interests span Information Visualization, Visual Analytics and Data Mining. He has published 70+ papers in refereed conferences and journals. He is the recipient of VAST Challenge Award in 2010 and 2012.

Zhiyang Teng is currently a Ph.D. student at Singapore University of Technology and Design. He received his MSc degrees from University of Chinese Academy of Sciences and his BEng degree from Northeastern University, China. His research interests include neural sentiment analysis and deep learning models for language parsing.



Le Wang is currently a graduate student in the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. He holds B.S. (2014) degree from the College of Software, Shandong University. His research interests include information visualization and visual analytics.

Yue Zhang is currently an assistant professor at Singapore University of Technology and Design. Yue Zhang received his DPhil and MSc degrees from University of Oxford, UK, and his BEng degree from Tsinghua University, China. His research interests include natural language processing, machine learning and artificial Intelligence. Yue Zhang serves as the associate editor for ACM Transactions on Asian and Low Resource Language Information Processing. He is the area chairs of COLING 2014, NAACL 2015, EMNLP 2015, ACL 2017 and EMNLP 2017, the TPC chair of IALP 2017.

Alexander (Alex) Binder is assistant Professor at SUTD Singapore. He obtained a Ph.D. degree at the department of computer science, Technical University Berlin in 2013. Since 2007, he has been working for the THESEUS project on semantic image retrieval at Fraunhofer FIRST. From 2010 to 2015 he was with the Machine Learning Group at the TU Berlin. His research interests include computer vision, medical applications, machine learning, efficient heuristics and understanding non-linear predictions.