Deep Learning for Event-Driven Stock Prediction

(based on the paper by X. Ding, Y. Zhang, T. Liu, J. Duan, 2015)

Kovalev Evgeny

Higher School of Economics, Faculty of Mathematics

Moscow, 2018

History

- Predicting stock price movements is a very important task for investors, public companies and governments
- But how can it be predicted?
- 1973, Burton Malkiel: The Random Walk Theory
    - prices are determined randomly
    - it is impossible to outperform the market
- AI: no, it is actually possible!
    - recent work: applying NLP techniques
    - result: events reported in financial news are important evidence to stock price movement prediction

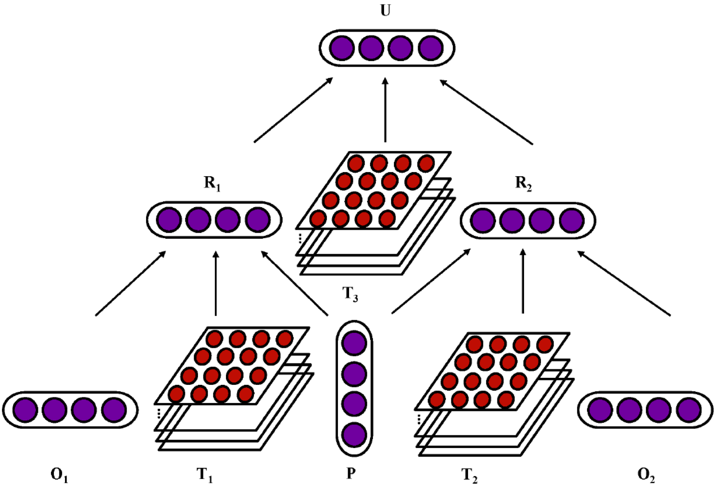
NLP techniques for financial news exploration

- simple features for news documents representation
  - bags-of-words
  - noun phrases
  - named entities
  - problem: structured relations are not captured
    - «Microsoft sues Barnes & Noble»
    - who accuses, who defends?
- structured representations of events
  - Open IE
    - «Microsoft sues Barnes & Noble»
    - (Actor = Microsoft, Action = sues, Object = Barnes & Noble)
  - problem: increased sparsity
- event embeddings — dense vectors
  - Neural Tensor Network for training event embeddings
  - Convolutional Neural Network for prediction

Event representation and extraction

- event representation: $E = \left(O_1, P, O_2, T\right)$
  - O_1 — actor
  - P — action
  - O_2 — object
  - T — timestamp (used for aligning stock data with news data)
- event extraction: Open IE technology and dependency parsing
  - ReVerb: extracting the candidate tuples of the event $\left(O_1', P', O_2\right)$
  - ZPar: extracting the subject, object and predicate
  - keeping only the tuples where O_1', O_2' and P' contain the subject, object and predicate, respectively

Neural Tensor Network



Neural Tensor Network

- Input: word embeddings, output: event embeddings
  - skip-gram algorithm for learning the initial word representation
  - actor, action and object representations = average of their word embeddings
- $R_1 = f\left(O_1^T T_1^{[1:k]} P + W \begin{bmatrix} O_1 \\ P \end{bmatrix} + b\right) \in \mathbb{R}^d$
  - $T_1^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ — a tensor
  - $O_1^T T_1^{[1:k]} P = r \in \mathbb{R}^k,\ r_i = O_1^T T_1^{[i]} P,\ i = 1, \dots, k$
  - other parameters are a standard feed-forward neural network
    - $W \in \mathbb{R}^{k \times 2d}$ — the weight matrix
    - $b \in \mathbb{R}^k$ — the bias vector
    - $f = \tanh$ — the activation function
- R_2 and U are computed in the same way as R_1
- pre-trained word embeddings give slightly better results than randomly initialized embeddings

Training

- training data: more than 10 million events from Reuters and Bloomberg financial news are extracted
- corrupted event tuple: $E^r = \left(O_1^r, P, O_2\right)$ (each word in O_1 is replaced with a random word from the training data dictionary)
- margin loss:

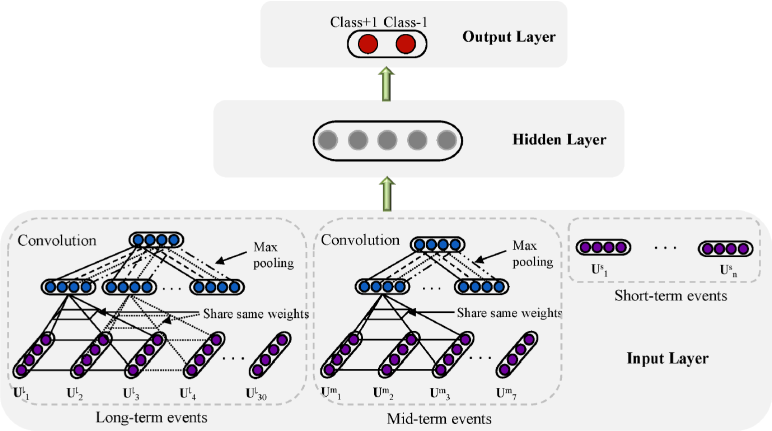
$$loss\left(E, E^r\right) = \max\left(0, 1 - f\left(E\right) + f\left(E^r\right)\right) + \lambda\|\Phi\|_2^2,$$

  where $\Phi = \left(T_1, T_2, T_3, W, b\right)$ is the set of parameters, $\lambda = 0.0001$ is the standard L_2 regularization weight
- if $loss\left(E, E^r\right) = 0$, the algorithm moves to the next event tuple
- otherwise, the parameters are updated to minimize the loss using the standard back-propagation algorithm

Deep Prediction Model

- Input: a sequence of event embeddings
  - long-term events: events over the past month
  - mid-term events: events over the past week
  - short-term events: events on the past day of the stock price change
  - events are arranged in chronological order
  - embeddings of the events on each day are averaged as a single input unit U
- Output: a binary class
  - class $+1$: the stock price will increase
  - class -1: the stock price will decrease

Deep Prediction Model



Architecture

- Convolution
  - used to combine $\ell = 3$ neighbour events
  - can be viewed as feature extraction based on sliding window
  - produces a new sequence Q:

$$Q_j = W_1^T U_{j-\ell+1:j},$$

where $U = (U_1, \ldots, U_n)$ is a series of input event embeddings, $U_i \in \mathbb{R}^d$, $W_1 \in \mathbb{R}^\ell$

- Max pooling
  - allows to determine the most representative features globally
  - produces a new sequence V:

$$V_j = \max Q(j, \cdot)$$

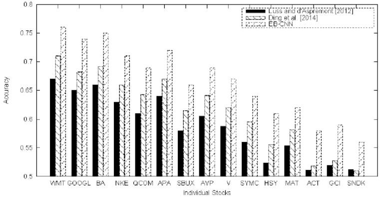
- Dense
  - feature layer: $V^C = (V^\ell, V^m, V^s)$
  - $y = \sigma\left(W_3^T \cdot \sigma\left(W_2^T \cdot V^C\right)\right)$

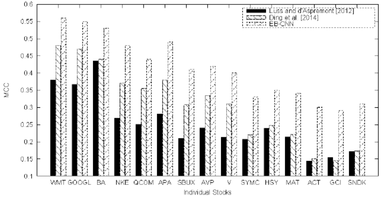
S&P 500 index prediction

	Acc	MCC
Luss and d'Aspremont (2012)	56.42%	0.0711
E-NN (Ding et al., 2014)	58.94%	0.1649
WB-NN	60.25%	0.1958
WB-CNN	61.73%	0.2147
E-NN	61.45%	0.2036
EB-NN	62.84%	0.3472
EB-CNN	65.08%	0.4357

- Luss and d'Aspremont: bags-of-words for news documents representation, SVM prediction model
- E — structured event tuples
- WB — word embeddings
- EB — event embeddings
- NN — standard NN prediction model
- CNN — CNN prediction model

Individual stock prediction





Market simulation

Stock	Profit of Lavrenko et al. [2000]	Profit of EBCNN
IBM	$47,000	$42,000
Lucent	$20,000	$27,000
Yahoo	$19,000	$32,000
Amazon	$14,000	$35,000
Disney	-$53,000	$7,000
AOL	-$18,000	$14,000
Intel	-$14,000	$8,000
Oracle	-$13,000	$17,000

Final results

	Index Prediction		Individual Stock Prediction		
	Acc	MCC	Acc	MCC	Profit
Luss [2012]	56.38%	0.07	58.74%	0.25	$8,671
Ding [2014]	58.83%	0.16	61.47%	0.31	$10,375
EB-CNN	64.21%	0.40	65.48%	0.41	$16,774

Conclusion

- Deep Learning is useful for event-driven stock price movement prediction!
  - Neural Tensor Network was proposed for learning event embeddings
  - deep Convolutional Neural Network was used to model the combined influence of long-term events and short-term events on stock price movements
- Event embeddings outperform discrete events-based methods
- Deep CNN can capture longer-term influence of news event than standard feedforward NN
- In market simulation, a simple greedy strategy allowed the proposed model to yield more profit compared with previous work

Thank you for your attention!