Mining Evidences for Concept Stock Recommendation

Qi Liu and Yue Zhang
Singapore University of Technology and Design,
8 Somapah Road, Singapore 487372
{qi_liu, yue_zhang}@sutd.edu.sg

Abstract

We investigate the task of mining relevant stocks given a topic of concern on emerging capital markets, for which there is lack of structural understanding. Deep learning is leveraged to mine evidences from large scale textual data, which contain valuable market information. In particular, distributed word similarities trained over large scale raw texts are taken as a basis of relevance measuring, and deep reinforcement learning is leveraged to learn a strategy of topic expansion, given a small amount of manually labeled data from financial analysts. Results on two Chinese stock market datasets show that our method outperforms a strong baseline using information retrieval techniques.

1 Introduction

Stock prices are affected by events. For example, recent announcement of a state plan to build a new economic region, Xiong'an near Beijing, by the Chinese government has led to the rise of hundreds of stocks, which can directly or indirectly benefit from the plan. As a second example, the winning of a lawsuit against IP (Intellectual Property) breach can strengthen investors' confidence on technological and entertainment companies. We refer to the topics or themes of such events (e.g. Xiong'an and IP) as *concepts* and their relevant stocks as *concept stocks*. Given a news event, it can be highly useful for investors to find a list of relevant concept stocks for making investment decisions.

For popular concepts, lists of relevant concept stocks can be found from analyst reports from financial websites. On the other hand, concepts are dynamic and flexible. In addition, insights can be relatively scarce for emerging capital markets, such as the Chinese market, which had been closed to foreign investments before 2015. It is therefore a challenging research question how to automatically find out potentially relevant stocks given a topic of interest, from a large market of multi-thousand equities.

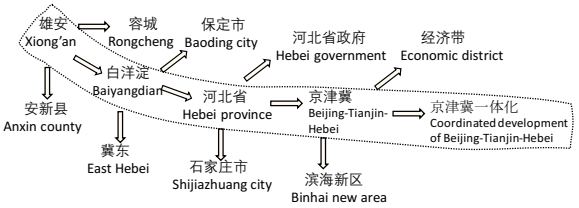


Figure 1: Concept relatedness. $A \Rightarrow B$ indicates that B is one of the 3 most related concepts to A.

Intuitively, evidences between concepts and stocks exist in text documents over the Internet. For example, news articles report events and companies involved. In addition, company filings such as annual/quarter reports contain factual knowledge about stocks, which can also be useful background information. For example, knowing that a company invests heavily on research is useful for correlating the company with IP-protection laws. Such evidence-mining process can involve multiple steps. As shown in Figure 1, starting from concept, Xiong'an, one might learn that the new economical region is located in the Baiyangdian area, which is further located in Hebei province. By further reading, one can infer that the new economic region is related with the coordinated development plan for the Beijing-Tianjin-Hebei region, and therefore benefit a wider range of stocks.

Based on the intuition above, we build a neural model for mining evidences for concept stock recommendation. The basis of our model is distributed similarities between concepts and stocks, obtained from embeddings trained over large-scale raw documents. Embedding similarities encode correlations from direct narrative evidence within context windows. To further include a multi-step

evidence mining, we build an iterative model for *concept expansion*, augmenting a given concept by iteratively adding more relevant concepts from background documents. As demonstrated in Figure 1, this process can be ambiguous, since there can be multiple directions for further reading given a set of concepts. We leverage a small amount of manually labeled data, downloaded from financial analysis websites, for guiding evidence mining.

In particular, we take a reinforcement learning method, which regards the evidence mining process as a decision process. The starting point is a given input concept, such as Xiong'an or Electronic Vehicle. At each step, a decision is made to stop further reading, or to continue adding related concepts to the set of concepts being considered. Existing concepts can also be removed from further consideration. Documents that discuss each concept are used to support the decision. After the process stops, relevant stocks to the set of concepts are recommended. The decision process is guided using a neural network model structure, trained with a loss function over the quality of the finally recommended stocks.

Results on two Chinese datasets show that our method outperforms a strong ranking-based baseline, which utilizes only direct evidences. Our method can be easily adapted for other markets given the availability of a small amount of training data. Our code is released[1].

2 Related Work

Our work is related to information retrieval and query expansion, where a concept can be regarded as a query and relevant stocks can be regarded as retrieved results. We rely on external evidence for correlating concepts and stocks.

Ranking is an important problem in information retrieval. We focus on ranking using neural models here. One line of work (Shen et al., 2014a,b) models queries and documents using convolutional neural network and ranks the documents pair-wise or list-wise. Another related method (Cao et al., 2015) adopts recursive neural networks to rank sentences for multi-document summarization. These methods requires massive annotated data, which is expensive to obtain for concept stock recommendation.

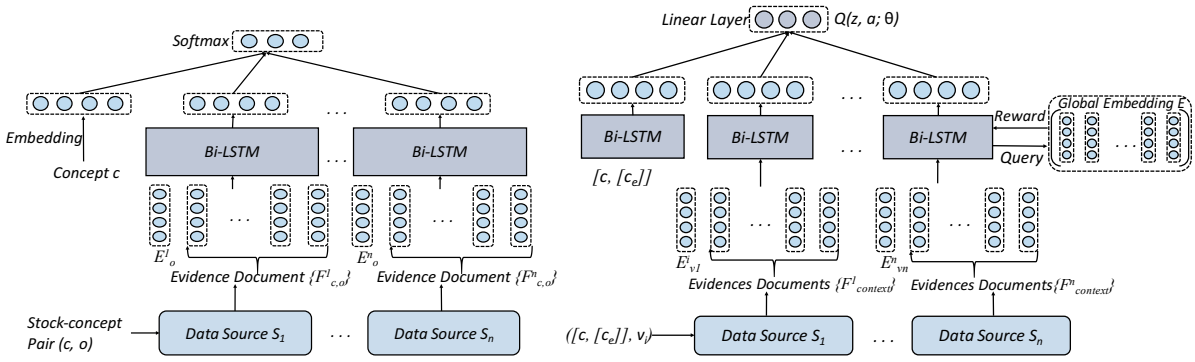
Query Expansion: One line of work (Cao et al., 2008; Preston and Colman, 2000) utilizes a feedback-based relevance model to expand queries. Another line applies language modeling to estimate conditional probabilities of concepts given a query, and expands the query with the most probable concepts (Bai et al., 2005; Carpineto and Romano, 2012). Recently, word embeddings are adopted for query expansion (Kuzi et al., 2016; Diaz et al., 2016). Our framework belongs to this line of work with a difference that we use reinforcement learning to dynamically expand queries instead of following handcrafted rules such as using k-nearest neighbors.

Reinforcement Learning: Our work aligns with existing work using reinforcement learning to collect evidences. Narasimhan et al. (2016) utilize external evidence to improve information extraction. While the work requires handcrafted features, our model uses dense embedding features. Athukorala et al. (2016) devise an interactive search engine balancing exploration and exploitation. Their work relies on user interaction to make decisions. In contrast, our work does not rely on active feedback, which can be expensive to obtain under our settings. Rodrigo and Cho (2017) introduce a query reformulation system based on reinforcement learning that rewrites a long and complex query to maximize the number of relevant documents returned. Differently, we do not assume complex queries and focus on recommending relevant stocks in our system. Zhong et al. (2017) solves a different problem, i.e. translating natural language questions to corresponding SQL queries.

3 Problem Definition

Our task is to find stocks relevant to a concept according to a variety of data sources, such as news, tweets and company files. Formally, given a concept c, a set of m stocks $\{o_i\}_{i=1}^{m}$ and n data sources $\{S_i\}_{i=1}^{n}$, where each S_i is a set of documents $\{D_j^i\}_{j=1}^{|S_i|}$ and each D_j^i is a sequence of words $w_1, w_2 ... w_{|D_j^i|}$, we assume the relevant stocks of the concept are revealed in the data sources (e.g. we discover PetroChina as a concept stock of 'petroleum' from the document 'PetroChina acquires Keppel's entire stake in Singapore Petroleum') and the task is to automatically discover these relations and select a subset of stocks as c's concept stocks based on the data sources $\{S_i\}_{i=1}^{n}$.

[1] https://github.com/leuchine/concept-stock-recommendation



(a) Ranking Baseline (b) Reinforcement Learning for Query Expansion

Figure 2: Concept Stock Recommendation Models

4 Representation

Motivated by the success of embedding-based models (Mikolov et al., 2013; Pennington et al., 2014) in capturing semantic regularities, we use embeddings to represent concepts, stocks and documents.

In particular, we adopt Chinese word segmentation (Yang et al., 2017) to obtain words from documents. Doc2Vec (Le and Mikolov, 2014) is then used on the documents of each data source S_i to obtain a local word embedding matrix E^i and a local document embedding matrix F^i, where each column of E^i (F^i) corresponds to a word (document) vector representation of S_i. In particular, we use embeddings, E^i_c and E^i_o as the local concept representation of c and the local stock representation of o in data source S_i, respectively. Furthermore, we obtain a global word embedding matrix E by averaging the local embedding matrices, $E^1 ... E^n$, where E_c and E_o are regarded as the global concept representation of c and the global stock representation of o, respectively.

We propose a ranking baseline and a reinforcement learning model for query expansion based on these representations.

5 Ranking Baseline

Inspired by Shen et al. (2014a; 2014b), our ranking baseline discriminatively projects the representations of concepts and evidences of stocks into a semantic space for measuring their relevances.

Mining Evidences: Formally, given a concept c and a stock o, we consult the data sources, retrieving the set of documents $\{D^i_{c,o}\}$ most relevant to (c, o) from each data source S_i as evidences.

To obtain evidences, we use c's local embedding E^i_c and o's local embedding E^i_o for representing the stock-concept pair (c, o). Cosine similarities are calculated between $E^i_c + E^i_o$ and each column of F^i for measuring the semantic relatedness of each document to (c, o). Suppose that the columns are normalized, the scores are calculated as:

$$score(\{D^i_j\}_{j=1}^{|S_i|}) = (F^i)^T(E^i_c + E^i_o) \quad (1)$$

q (q is set as 5 empirically) documents $\{D^i_{c,o}\}$ with the maximum scores are selected as evidences from each S_i. When $|F^i|$ is large, we use approximate k-nearest-neighbor algorithms, namely Locality Sensitive Hashing (Datar et al., 2004), to improve efficiency.

Learning to Rank o Given c: The overall framework for measuring relevances is shown in Figure 2 (a). Given a concept c and stock o, for each data source S_i, the local stock representation E^i_o and the local document representations of the q most relevant documents, denoted as $\{F^i_{c,o}\}$, are sequentially fed into Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to acquire a semantic representation of the evidences.

A bidirectional extension (Graves and Schmidhuber, 2005) is applied to capture semantics both left-to-right and right-to-left. As a result, two sequences of hidden states are obtained, i.e. $\overrightarrow{h}_1, \overrightarrow{h}_2 ... \overrightarrow{h}_{q+1}$ and $\overleftarrow{h}_1, \overleftarrow{h}_2 ... \overleftarrow{h}_{q+1}$. We concatenate $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ at each time step to obtain the final hidden states $h_1, h_2 ... h_{q+1}$.

Average pooling (Boureau et al., 2010) is applied on the hidden states $h_1, h_2 ... h_{q+1}$ to obtain

evidence representation $I^i_{c,o}$,

$$I^i_{c,o} = \frac{\sum_{t=1}^{q+1} h_t}{q+1} \quad (2)$$

We concatenate all $I^i_{c,o}$ with the global concept representation of c (i.e. the average of local representations, $E^1_c ... E^n_c$) and feed the result to a softmax layer to obtain the probability of a stock o being c's concept stock, denoted as $p(o|c)$. Given a concept c, all stocks are ranked by the probabilities.

Given a set of gold-standard concept stock data, supervised learning is conducted to learn $p(o|c)$. The loss function is defined as:

$$E_{(c,o,y)}[-log\, p(o|c)^y - log\, (1-p(o|c))^{(1-y)}] \quad (3)$$

Here y is 1 when o is a concept stock of c, and 0 otherwise. Equation 3 maximizes $p(o|c)$ $(1 - p(o|c))$ when $y = 1$ $(y = 0)$. AdaGrad (Duchi et al., 2011) is applied to update parameters.

6 Recommendation via Query Expansion

The ranking baseline can require large amounts of annotated data to deliver satisfying performance (Shen et al., 2014a,b), which can be costly. In addition, the algorithm has to deal with highly imbalanced datasets, since there are thousands of stocks in a stock market, but only a few are related to a concept c, which greatly harms the performance of discriminatively trained algorithms (Wu and Chang, 2003).

We take a different approach, utilizing the same data sources and representations as the ranking baseline. To better leverage a small amount of supervision data, we apply reinforcement learning to expand the query concept c, consulting supporting evidences from the data sources $\{S_i\}^m_{i=1}$. We leverage embedding similarities as a basis for concept-stock relatedness. The advantage is that embeddings can be trained over large scale raw texts unsupervisedly, without the need for manually labeled stock lists.

6.1 Direct Semantic Relatedness

Embeddings represent similarities between concepts and stocks if they co-exist literally in a context window during embedding training. As a result, irrelevant (relevant) stocks are less (more) similar to the concept c, since they infrequently (frequently) co-occur, which alleviates the problem brought by imbalanced datasets in that irrelevant stocks can be spotted at ease.

Global representations of c and o is utilized to obtain a direct relevance score $\hat{f}(c, o)$:

$$\hat{f}(c, o) = E_c \cdot E_o, \quad (4)$$

where $\cdot$ denotes the dot product operation. The stocks are ranked by $\hat{f}(c, o)$ and concept stocks are these with the maximum cosine similarities.

6.2 Indirect Semantic Relatedness by Query Expansion

While $\hat{f}(c, o)$ measures direct relevance between c and o in embedding contexts, we want to find those o' that are indirectly relevant to c by reasoning as shown in Figure 1. Query expansion (Kuzi et al., 2016; Diaz et al., 2016) is used to this end. One naive baseline is expanding the concept c with its k-nearest-neighbor concepts, denoted as $[c_e]$, from global matrix E measured by cosine similarity. Relevance between the expanded concepts $[c, [c_e]]$ and o is calculated as:

$$\begin{aligned}\hat{f}([c, [c_e]], o) &= E_{[c,[c_e]]} \cdot E_o \\ &= (E_c + \sum_{c_e \in [c_e]} E_{c_e}) \cdot E_o \end{aligned} \quad (5)$$

We define $E_{[c,[c_e]]}$ as the addition of E_c and each E_{c_e}. The baseline is relatively inflexible since a fixed number of k expansion concepts are selected for all c. In contrast, the reasoning procedures shown in Figure 1 can take an arbitrary number of steps. Besides, the naive baseline does not incorporate supervision, thus being unable to decide whether the selected concepts are beneficial for concept stock recommendation.

We use reinforcement learning to tackle this issue, directly learning how to expand queries from a few labeled cases. Given c, our method works iteratively, expanding the concept until it expects further expansions are not desired. For each candidate concept to expand c, a decision is made by the model on whether it will improve, worsen or have no effect on recommendation accuracies.

Based on these, we model query expansion with a Markov Decision Process (MDP) to discriminatively select expansion concepts for c to maximize recommendation accuracies, while requiring much less training data compared to the ranking baseline.

The overall framework is shown in Figure 2 (b). Formally, a MDP is a list $[Z, A, T, R]$, where $Z = \{z\}$ is a set of states, $A = \{a\}$ is a set of actions, $T(z, a)$ is a transition function, which determines the next state $z' = T(z, a)$ after performing action a on z, and R is a reward function. We describe each in detail below:

States: Each state z is a list of lists:

$$z = [[c, [c_e]], [v^1, \{F^1_{context}\}]...[v^n, \{F^n_{context}\}]], \quad (6)$$

where $[c, [c_e]]$ consists of the input concept c and its expansion concept list $[c_e]$ so far. In the start state, $[c_e]$ is empty, and thus $[c, [c_e]] = [c, [\,]]$. $[v^i, \{F^i_{context}\}]$ consists of a new candidate concept v^i and its supporting evidences $\{F^i_{context}\}$. Since globally trained embeddings underperform locally trained embeddings (Diaz et al., 2016) for query expansion, we use local embeddings E^i to suggest candidate concepts v^i, instead of using global embedding E. v^i is obtained by finding the most similar concept of $[c, [c_e]]$ from local embeddings E^i. $\{F^i_{context}\}$ is the document representations of the q most relevant documents to $([c, [c_e]], v^i)$ as evidences. Formally, $\{F^i_{context}\}$ is the document representations of the q documents with the maximum scores, $score'(\{D^i_j\}^{|S_i|}_{j=1}) = (F^i)^T(E^i_{[c,[c_e]]} + E^i_{v^i})$. As a result, at each state, we have n candidate concepts $v^1...v^n$. The neural agent chooses at most one concept to be added to $[c_e]$ based on the evidences.

Action: The agent can take four types of actions. (1) Add one of n candidate concepts to $[c_e]$ (2) Reject all n candidate concepts. (3) Remove the last added concept from $[c_e]$ (4) Stop the process.

State Transition: After taking an action a on a state z, a new state z' is yielded by the transition function $T(z, a)$. If one of the candidate concepts v^i is chosen, v^i is added to $[c_e]$. In addition, the new state z' is obtained by updating $[v^1, \{F^1_{context}\}]...[v^n, \{F^n_{context}\}]$ by finding the most similar concepts of the new $[c, [c_e]']$ and their supporting evidences among the local embeddings. If action (2) is chosen, $[c, [c_e]]$ remains, while the $v^1...v^n$ is replaced with the second most similar concepts of $[c, [c_e]]$ among the local embeddings, and the process repeats until action (1), (3) or (4) is chosen. If (3) is chosen, the last added concept is removed from $[c_e]$, and $[v^1, \{F^1_{context}\}]...[v^n, \{F^n_{context}\}]$ are updated according to the new $[c, [c_e]']$. If (4) is chosen, the query expansion process finishes. The final $[c, [c_e]]$ is the result of query expansion.

Neural Agent: Given a state z, the neural agent chooses one action to take among the four types of actions. To this end, $[c, [c_e]]$ and each $[v^i, \{F^i_{context}\}]$ are fed into separate Bi-LSTM to obtain a concept representation and candidate concept representations, respectively. We further concatenate these representations and use a linear layer to obtain Q-values $Q(z, a; \theta)$ for each action a (Sutton and Barto, 1998). Note that we do not use softmax to normalize the Q-values since Q-value is the expectation of discounted sums of rewards by definition instead of probabilities. The action with the maximum Q-value is chosen.

Reward: A reward r is associated at each step specified by the reward function R, which evaluates the goodness of action a on state z. We use the difference of mean average precision (MAP) (Christopher et al., 2008) before and after an action a as the reward function:

$$R(z, a, z') = MAP(z') - MAP(z), \quad (7)$$

where MAP is defined as:

$$MAP(z) = \frac{1}{|\omega(c)|} \sum_{o \in \omega(c)} Precision@rank(o; z, E) \quad (8)$$

and

$$Precision@K = \frac{\sum_{o' \in \omega(c)} \mathbb{1}(rank(o'; z, E) \leq K)}{K} \quad (9)$$

$\omega(c)$ is the set of concept stocks of the concept c in training data. $rank(o; z, E)$ is the rank of the stock o, which is calculated by utilizing $[c, [c_e]]$ of z and global embedding E to rank all stocks using Equation 5. $\mathbb{1}$ is the indicator function. Therefore, MAP measures the goodness of the ranking, which is large if the stocks in $\omega(c)$ are ranked higher compared to the others. Reward r is positive if $[c, [c_e]']$ of z' can rank stocks better compared to $[c, [c_e]]$ of z and negative otherwise.

We choose MAP based on two reasons: (1) MAP provides a measure of quality, which has been shown to have good discrimination and stability. Besides, MAP is roughly the average area under the precision-recall curve for a set of queries (Christopher et al., 2008). Thus, optimizing MAP can indirectly improve both precision and recall. (2) MAP provides smoother scores than other metrics such as Precision@K and Recall@K.

In summary, at each step, the MDP framework chooses an action a based on a state z, obtaining a

Algorithm 1 Training Phase of MDP for Query Expansion

1: Initialize experience memory M
2: Initialize action network with random weights θ
3: Initialize target network with weights $\theta_{target} \leftarrow \theta$
4: **for** episode from 1 to N **do**
5: **for** each concept c **do**
6: Obtain start state $z \leftarrow$*get_state*$([c, [\,]], E_1 ... E_n)$
7: **while** $true$ **do**
8: **if** $random() < \epsilon$ **then**
9: Select a random action a
10: **else**
11: Send state z to neural agent
12: Obtain action a from action network
13: **end if**
14: Obtain new state $z' \leftarrow T(z, a)$
15: Calculate reward, $r \leftarrow R(z, a, z')$
16: Store sample (z, a, z', r) to M
17: Update state $z \leftarrow z'$
18: Sample mini-batch (z_t, a_t, z'_t, r_t) from M
19: Calculate sample estimate using Equation 11
20: Perform a batch gradient descent step on the action network, updating parameters θ using Equation 12
21: Update $\theta_{target} \leftarrow \theta$ at every C steps.
22: **if** $a == action\ (4)$ **then**
23: **break**
24: **end if**
25: **end while**
26: Send the final $[c, [c_e]]$ to E and rank the stocks
27: **end for**
28: **end for**

new state z' and a reward r, which forms a sample, (z, a, z', r). The process repeats until action (4) is chosen.

6.3 Learning

We adopt Q-learning (Sutton and Barto, 1998) to optimize the neural agent, which uses a function $Q(z, a)$ to represent Q-values and the recursive Bellman equation to perform Q-value iteration, when observing a new sample (z, a, z', r). Since the state space Z can be extremely large in practice, we represent the Q-value function $Q(z, a)$ with a neural agent shown in Figure 2 (b) named the action network parametrized by θ (Mnih et al., 2015). The deep Q-learning method has the ability to capture nonlinear features and achieve better performance compared with traditional methods (Narasimhan et al., 2015). Formally,

$$Q(z, a) = Q(z, a; \theta) \tag{10}$$

To improve learning stability, sample reward estimates are obtained from a separate target network with the same architecture as the action network (Mnih et al., 2015), parametrized by θ_{target}. Formally, the sample reward estimate of (z, a, z', r) is:

$$y' = \begin{cases} r & if\ a = action(4) \\ r + \gamma \max_{a_{new} \in A} Q(z', a_{new}; \theta_{target}) & otherwise \end{cases} \tag{11}$$

Note that if the action (4) is taken, $y' = r$ since the process stops at state z and no further action will be taken so that the sum of rewards is r.

To learn the model parameters θ, the action network outputs $Q(z, a; \theta)$ should be close to sample estimates obtained from target network. Thus, we introduce an experience memory M to save history samples and select a mini-batch of samples according to a uniform distribution. We use the mean square error as the loss function:

$$E_{(z,a,z',r) \sim U(M)}[(Q(z, a; \theta) - y')^2] \tag{12}$$

The training phase is shown in Algorithm 1. In lines 8-12, we use ϵ-greedy exploration, which encourages the agent to explore unknown state space (Sutton and Barto, 1998).

7 Experiments

7.1 Datasets

We construct two datasets from the Chinese websites, Jinrongjie[2] and Tonghuashun[3], respectively, which are two mass medias for China stock markets. These two websites periodically publish their concept stock lists, which are manually collected and analyzed by their financial professionals. We observe high quote change correlations of the stocks of each concept c and their lists are commonly used by investors to select stocks, which confirms the credibility of these datasets. The Jinrongjie dataset consists of 206 concepts and each concept has an average of 25.4 manually suggested concept stocks. For the Tonghuashun dataset, there are 900 concepts and 15.6 manually suggested concept stocks on average.

There are two main stock exchanges in China, the Shanghai Stock Exchange[4] and the Shenzhen Stock Exchange[5]. We crawled stock lists from their official websites, with 3326 stocks in total.

We utilize four public data sources, S_1 to S_4, the statistics of which are shown in Table 1.

[2]金融界 http://stock.jrj.com.cn/concept/
[3]同花顺 http://stock.10jqka.com.cn/gngyw_list/
[4]上海证券交易所 http://www.sse.com.cn/assortment/stock/list/share/
[5]深圳证券交易所 http://www.szse.cn/

Source	# Docs	Avg # Words
News	255,318	2753
Report	12, 431	19,145
Wikipedia	2,143	3745
Search Engine	6,130	1846

Table 1: Data Source statistics

S_1: *News* is crawled from Sina Finance News[6], which originates from 2009 to 2017.

S_2: *Reports* consists of annual and quarterly company reports crawled from Sina Finance[7].

S_3: *Wikipedia* includes relevant wikipedia pages of the concepts and stocks, if any, which can provide some background knowledge.

S_4: *Search Engine* includes open-domain information for the concepts and stocks obtained using Bing API [8]. We adopt search engine results for representing heterogeneous web texts. The top-ranked webpages are crawled.

Given the Jinrongjie and Tonghuashun datasets, we randomly select 70%, 10% and 20% of the concepts as training, development and testing sets, respectively.

7.2 Baselines and Parameter Settings

We compare our method with four baselines:

Search is a naive information retrieval baseline, which sends the concept c and each stock o to an inverted index and obtains a list of top-k ranked documents ($k = 5$ in experiments) by a fixed ranking metric, Ocapi BM25 (Robertson et al., 2009). The stocks are ranked by the average of top-k documents' BM25 scores.

Rank is our ranking baseline. Five top-ranked documents from each source are fed into the model. All 3326 stocks are ranked for each concept.

Semantics ranks the stocks using Equation 4, which is the naive semantic relatedness $\hat{f}(c, o)$.

Semantics+ extends **Semantics** by including 8 most similar words to expand original concepts.

Semantics++ extends **Semantics** by including the most similar words with similarities larger than 0.65 to expand original concepts. On average, 6.3 concepts are included.

[6] http://finance.sina.com.cn/

[7] http://finance.sina.com.cn/focus/ssgsnb2016/

[8] https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api/

Jinrongjie				
Method	$P@5$	$P@10$	$R@30$	MAP
Search	0.402	0.315	0.338	0.296
Semantics	0.45	0.367	0.380	0.332
Semantics+	0.471	0.370	0.391	0.352
Semantics++	0.478	0.375	0.396	0.359
Rank	0.467	0.376	0.402	0.365
RL	**0.524**$*$	**0.427**$*$	**0.428**$*$	**0.398**$*$
Tonghuashun				
Method	$P@5$	$P@10$	$R@30$	MAP
Search	0.387	0.302	0.315	0.278
Semantics	0.437	0.347	0.360	0.327
Semantics+	0.448	0.356	0.374	0.345
Semantics++	0.453	0.362	0.380	0.351
Rank	0.458	0.373	0.381	0.356
RL	**0.507**$*$	**0.402**	**0.422**$*$	**0.378**$*$

Table 2: Concept stock recommendation results on Jinrongjie and Tonghuashun. $*$ denotes statistical significance using Wilcoxon signed rank test ($p < 0.05$)

For our model, denoted as **RL**, we set the window size as 80, the embedding size as 300 and the vocabulary size as $100,000$ in view of the large variety of phrases after word segmentation to train Doc2Vec. Also, the experience memory size is set to $50,000$ and older training samples are abandoned. The ϵ value is set as 1 at the start and gradually decreases to 0.1 after 3000 annealing steps. We perform a training phase after every 3 decision steps. The mini-batch size is set to 50. Dropout is applied to avoid overfitting and the dropout rate is 0.5. We set the learning rate for AdaGrad as 0.01. Gradient clipping (Pascanu et al., 2013) is adopted to prevent gradient exploding and vanishing during training process.

7.3 Recommendation Accuracies

We use four metrics, mean average precision (MAP), precision at 5 and 10 ($P@5$, $P@10$) and recall at 30 ($R@30$) to evaluate the algorithms. The results are shown in Table 2.

From Table 2, the first observation is that **RL** outperforms the baselines on both datasets, which demonstrates the effectiveness of combining semantic relatedness with query expansion based on reinforcement learning. The baseline **Rank** achieves the second best results. The large gap between **RL** and **Rank** indicates that **RL** is much easier to train compared to **Rank** on small data.

Second, we observe that **Semantics+** improves over **Semantics**, which shows that query expansion has the potentials to alleviate concept ambiguities and benefit concept stock recommendation. **Semantics++** can outperform **Semantics+** by considering semantic similarities. Also, compared to

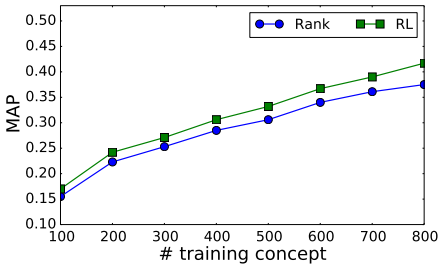


Figure 3: Learning Curve

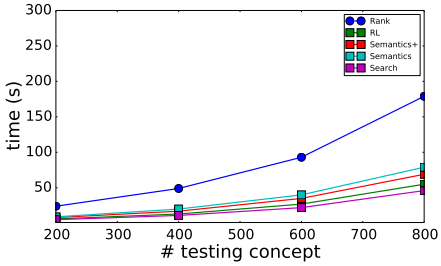


Figure 4: Efficiency

RL, we conclude that query expansion based on reinforcement learning could better utilize training data and significantly outperform naive query expansion methods.

The last observation is that **Search** performs the worst among the methods. This sheds light on the limitations of traditional search models and confirms the effectiveness of semantic modeling by word embedding and neural models.

7.4 Influence of Size of Training Data

We increase the amount of training concepts and study whether **RL** is easier to train than **Rank**. The results on Tonghuashun is shown in Figure 3 (similar patterns are demonstrated using Jinrongjie). With more training concepts, the MAPs of both methods increase. However, **RL** consistently outperforms **Rank** and the margin becomes larger. Thusly, we conclude that **RL** requires less data than **Rank** to achieve similar performance.

7.5 Efficiency Comparison

Figure 4 shows the efficiency of all algorithms on testing data. The three unsupervised algorithm **Search**, **Semantics** and **Semantics+** are more efficient compared to the supervised algorithm, **Rank** and **RL**. **RL** is more efficient compared to **Rank**, since **Rank** has to rank every stock to obtain concept stocks.

中字头 (Sino)		
Semantics+	*Rank*	*RL*
中远海控ZYHK	中国国航ZGGH	中国交建ZGJJ
中国交建ZGJJ	中国中铁ZGZT	中国中冶ZZZY
~~石油济柴SYJC~~	中国石油ZGSY	中国建筑ZGJZ
中国一重ZGYZ	~~中储股份ZCGF~~	中国中铁ZGZT
~~招商轮船ZSLC~~	中国中冶ZGZY	中国铁建ZGTJ
特斯拉 (Tesla)		
Semantics+	*Rank*	*RL*
~~万向钱潮WXQC~~	协鑫集成XXJC	~~万向钱潮WXQC~~
~~协鑫集成XXJC~~	~~比亚迪BYD~~	国机汽车GJQC
天汽模TQM	亚太股份YTGF	天汽模TQM
亚太股份YTGF	天汽模TQM	亚太股份YTGF
~~爱康科技AKKJ~~	~~长城汽车CCQC~~	~~上海临港SHLG~~
智能物流 (Intelligent Logistics)		
Semantics+	*Rank*	*RL*
东杰智能DJZN	~~华胜天成HSTC~~	飞力达FLD
~~亿阳信通YYXT~~	飞力达FLD	中储股份ZZGF
飞力达FLD	~~美菱电器MLDQ~~	东杰智能DJZN
~~美克家居MKJJ~~	东杰智能DJZN	圆通速递YTSD
~~天成自控TCZK~~	圆通速递YTSD	华鹏飞HPF

Table 3: Example concept stocks, where ~~stock~~ indicates a incorrectly recognized concept stock.

7.6 Case Study

Data Sources Effectiveness: To study the effectiveness of data sources, we count how many concepts are chosen from each data source during query expansion. For the Tonghuashun test data (similar tendencies are observed for Jinrongjie), 761, 689, 199, 344 concepts are selected for S_1-S_4, respectively. Accumulated rewards of these concepts for S_1-S_4 are 76.13, 61.32, 7.49 and 14.10, respectively. We conclude that *News* and *Reports* are relatively more effective for improving recommendation accuracies.

Recommended Stocks: To obtain a better understanding of our method, we examine the symbols of the top-5 selected stocks of concepts and some examples are shown in Table 3.

We notice that **RL** can effectively extend concepts with relevant concepts. For example, the algorithm extends 中字头 (Sino) with 中国 (China) and 国企 (State-owned enterprises), 特斯拉 (Tesla) with 入华 (into China), 电动车 (Electric cars) and 马斯克 (Musk) and 智能物流 (Intelligent Logistics) with 物流 (Logistics), CSN (China Smart Logistic Network), 仓储 (Warehousing) and 配送 (Delivery), which results in more accurate concept stocks.

For 特斯拉 (Tesla), **RL** made two mistakes due to rumor and ambiguity. For example, 上海临港SHLG is chosen because of rumors that Tesla will establish a new factory there. 万向钱潮WXQC is mistakenly chosen because 万向钱潮WXQC is called China's Tesla in some news due to its investments in electric cars. In contract, **Semantics+** and **Rank** are limited by lack of su-

pervision and highly unbalanced datasets, respectively. For example, **Rank** mistakenly chooses 美菱电器MLDQ in that it confuses 智能家电 (Smart Appliances) with 智能物流 (Intelligent Logistics). We conclude that **RL** is capable of expanding concepts with relevant concepts that helps find more revelant stocks.

Conclusion

We have investigated a reinforcement learning method to automatically mine evidences from large-scale text data for measuring the correlation between a concept and a list of stocks. Compared to standard information retrieval methods, our method leverages a small amount of training data for obtaining a flexible strategy of query expansion, thus being able to disambiguate contexts in exploration. Results on two Chinese datasets show that our method is highly competitive for our task, thus providing a tool for investors to gain understandings of emerging markets.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. We would like to thank Yumin Zhou for her insightful discussion and assisting coding. Yue Zhang is the corresponding author.

References

Kumaripaba Athukorala, Alan Medlar, Antti Oulasvirta, Giulio Jacucci, and Dorota Glowacka. 2016. Beyond relevance: Adapting exploration/exploitation in information retrieval. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*. ACM, pages 359–369.

Jing Bai, Dawei Song, Peter Bruza, Jian-Yun Nie, and Guihong Cao. 2005. Query expansion using term relationships in language models for information retrieval. In *CIKM*. ACM, pages 688–695.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010. A theoretical analysis of feature pooling in visual recognition. In *ICML*. pages 111–118.

Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting good expansion terms for pseudo-relevance feedback. In *SIGIR*. ACM, pages 243–250.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015. Ranking with recursive neural networks and its application to multi-document summarization. In *AAAI*. pages 2153–2159.

Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys* 44(1):1.

D Manning Christopher, Raghavan Prabhakar, and SCHÜTZE Hinrich. 2008. Introduction to information retrieval. *An Introduction To Information Retrieval* 151:177.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, pages 253–262.

Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. *arXiv preprint arXiv:1605.07891* .

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* pages 2121–2159.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* pages 602–610.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. MIT Press, pages 1735–1780.

Saar Kuzi, Anna Shtok, and Oren Kurland. 2016. Query expansion using word embeddings. In *CIKM*. ACM, pages 1929–1932.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*. pages 1188–1196.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. pages 3111–3119.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. 2015. Human-level control through deep reinforcement learning. *Nature* pages 529–533.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941* .

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. *arXiv preprint arXiv:1704.04572* .

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML* 28:1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Carolyn C Preston and Andrew M Colman. 2000. Optimal number of response categories in rating scales: reliability, validity, discriminating power, and respondent preferences. *Acta psychologica* pages 1–15.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval* pages 333–389.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014a. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014b. Learning semantic representations using convolutional neural networks for web search. In *WWW*. ACM, pages 373–374.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. MIT press Cambridge.

Gang Wu and Edward Y Chang. 2003. Class-boundary alignment for imbalanced dataset learning. In *ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC*. pages 49–56.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *ACL*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103* .