Semantic Feature Representation to Capture News Impact

First Author
Affiliation / Address line 1
Affiliation / Address line 2
email@domain

Second Author
Affiliation / Address line 1
Affiliation / Address line 2
email@domain

Abstract

Mining natural language text for real world applications can benefit from a data representation that encodes semantic structure. Tree-based features, however, limit the available learning methods. This paper presents a study where semantic frames are used to mine financial news so as to quantify the impact of news on the stock market. We represent news documents in a novel semantic tree structure and use tree kernel support vector machine to predict the change of stock price. We achieve an efficient computation through linearization of tree kernels. In addition to two binary classification tasks, we rank news items according to their probability to affect change of price using two ranking methods that require vector space features. We evaluate our rankings based on receiver operating characteristic curves and analyze the predictive power of our semantic features. For both approaches, the proposed semantic features provide superior results.

Introduction

The acquisition and analysis of information through daily news, annual reports, and other texts is crucial for decision making in financial activities. Providers of online financial news provide updates throughout the day monitored by traders and others. There are also many sources of quantitative financial data, such as corporate credit ratings, earnings, and share prices. Our general goal is to ground information derived from application of NLP techniques to financial news in such real world observations. Our previous work relied on a heterogeneous data representation that mixed BOW vectors and semantic trees, and relied on tree kernel learning methods. Here we convert the tree features to a linear representation to test the hypothesis that a more homogeneous representation can improve learning performance, and to recast the learning problem from classification to regression.

We label data instances derived from Thomson Reuters news using the daily price of publicly traded companies. In our previous work (Anon), comparison of many data representations demonstrated the advantage of including tree features based on semantic frame parsing. The motivation for the tree representation was threefold: (1) to generalize over different lexical items that evoke the same semantic frame, where a frame represents a general scenario; (2) to encode the semantic dependencies among words and phrases within a sentence, and (3) to extract distinct semantic roles and relations for each entity of interest (companies). To use tree features, we relied on tree kernel learning, where common tree fragments across trees are found by iterative tree traversal. The advantages accrued from tree features that encode semantic structure come at the cost of: (1) high computational complexity; (2) restriction to classification tasks; (3) lack of comparability of the tree features with the vector-based features we combined them with. Linearization of the tree features overcomes all three disadvantages.

A feature representation that combines linearized tree features with BOW and other vector features performs as well as, or better than, the heterogenous representation with semantic trees at two binary classification tasks: predicting whether a company's price changes, and predicting whether a price change is positive or negative. The homogeneous vector representation also makes it possible to use information derived from semantic frame parsing for ranking companies, and for regressing on price change. We first discuss related work and the dataset. The next section introduces our tree-based features derived from semantic frame parses, and the linearization method, followed by a section that presents two ranking algorithms used here. The next section describes the experimental design and results, followed by discussion and conclusions.

Related Work

Analysis of financial news for market analysis has drawn increasing attention. Kogan analyzed quarterly earning reports to predict stock return volatility and to predict whether a company will be delisted (2009). Luss and d'Aspremont used text classification to model price movements of financial assets on a per-day basis (2011). They tried to predict the direction of return and abnormal returns, defined as an absolute return greater than a predefined threshold. Schumaker and Chen proposed a stock price prediction system based on financial news. They represent documents by boolean valued bag-of-words and named entities (2010). Wang et al. presented a framework for mining the relationships between news articles and financial instruments using a rule-based expert system (2011). Most of the active research explores the financial instruments where mining news can be beneficial. However, none of these focuses on document representation with rich linguistic features, and they rarely go beyond a bag-of-words (BOW) model. Luss and d'Aspremont (2011) and Lavrenko (2000) both state the need for document feature engineering as future research directions. In this study, we explore a rich feature space with a novel document representation that relies on frame semantic parsing.

Our work is also related to sentiment analysis. We mine opinions about entities of interest, which later feeds a ranking model. Schumaker et al. treat stock price prediction as a sentiment analysis problem to distinguish positive and negative financial news (Schumaker et al. 2012). Tetlock et al. quantify pessimism of news using General Inquirer (GI), a content analysis program (2007) and (2008). Feldman applies sentiment analysis on financial news using rule-based information extraction and dictionary-based prior polarity scores (2011). In this study, our model addresses a fine-grained sentiment analysis task that distinguishes different entities mentioned in the same sentence, and their distinct roles in sentiment bearing semantic frames.

Data and Labels

Data

Publicly available Reuters news from 2007 through 2012 is used for this study. This time frame includes a severe economic downturn in 2007-2010 followed by a modest recovery in 2011-2012, which makes our task particularly challenging. An information extraction pipeline is used to preprocess the data. The timestamp of the news is extracted for later alignment with stock price information, as explained below. A company mention is identified by a rule-based matching of a finite list of companies from the S&P 500[1]. On the assumption that the news within the same industry will be more homogeneous, we focus on one sector, consumer staples, which is one of the largest of N sectors. Each data instance in our experiment is a tuple of a consumer staples company and the sentence that mentions it. The data contains 40 companies, such as Walmart and Proctor & Gamble, and 32,311 news articles with 91,177 sentences. Each relevant sentence mentions 1.07 companies on average, which adds up to 97,559 data instances.

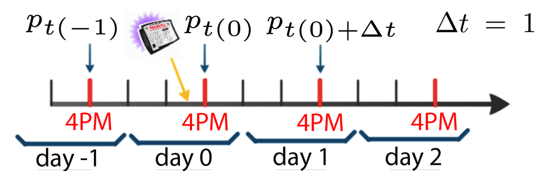
Aligning Price Labels with News

We align publicly available daily stock price data from Yahoo Finance with the Reuters news using a method to avoid back-casting, as illustrated in Figure 1. Two kinds of labels are assigned: the existence of a price change and the direction of change. The *change* in price and *polarity* tasks are each treated as binary classification problems. Based on the finding of a one-day delay of the price response to the information embedded in the news by (Tetlock, Saar-Tsechansky, and Macskassy 2008), we use $\Delta t = 1$ in our experiment, as illustrated in Figure 1. To constrain the number of parameters, we also use a threshold value (r) of a 2% change, based on the distribution of price changes across our data. In future work, this could be tuned to sector or time.

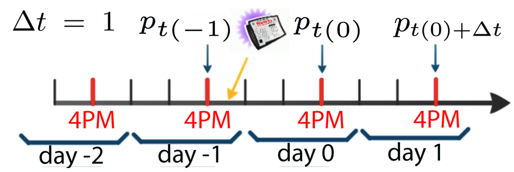
$$\text{change}=\begin{cases} +1 & \text{if } \frac{|p_{t(0)+\Delta t} - p_{t(-1)}|}{p_{t(-1)}} > r \\ -1 & \text{otherwise} \end{cases}$$

$$\text{polarity}=\begin{cases} +1 & \text{if } p_{t(0)+\Delta t} > p_{t(-1)} \text{ and } change = +1 \\ -1 & \text{if } p_{t(0)+\Delta t} < p_{t(-1)} \text{ and } change = +1 \end{cases}$$

The labels are derived from the daily closing price - the price quoted at the end of a trading day (4PM US Eastern Time). $p_{t(-1)}$ is the adjusted closing price at the end of the last trading day, and $p_{t(0)+\Delta t}$ is the price of the end of the trading day after the Δt day delay. Only the instances witha price change are included in the *polarity* task.



(a) News during trading hours



(b) News after trading hours

Figure 1: Aligning stock price with news.

Data Representation

We present a document representation based on frame semantics. Each data instance is encoded in a tree structure, referred to as a SemTree, constructed from semantic parses, with a given designated object (a company) promoted to the root node. The children are the object's semantic roles. In the last part of this section, we also introduce other features included in our representation.

SemTrees

FrameNet (Baker, Fillmore, and Lowe 1998) provides frame-semantic descriptions of thousands of English lexical items based on the theory of semantic frames (Fillmore 1976). There have been many studies of semantic frame parsing. The parses we use are derived from SEMAFOR[2] (Das and Smith 2012), which solves the semantic parsing problem by rule-based identification of targets (lexical units that trigger frames), and a log-linear model for frame identification and frame element filling.

Figure 2 shows an example frame tree for the sentence *Oracle sued Google*, where Judgment_communication is the frame (F) triggered by target (T) *sue*, with text span *Oracle* being the COMMUNICATOR element, and *Google* being the EVALUEE element.

[1]The Standard & Poor's 500 is an equity market index that includes 500 U.S. leading companies in leading industries.

[2]http://www.ark.cs.cmu.edu/SEMAFOR.

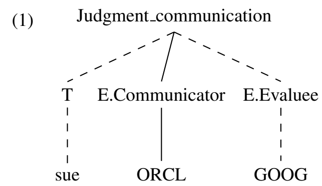


Figure 2: Frame tree for *Oracle sued Google*.

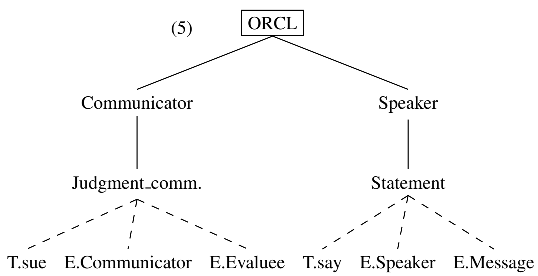


Figure 3: Constructing a SemTree for *Oracle* from the sentence *Oracle sued Google, saying Google infringes its patents for the Java programming language.*

Multiple frames can be triggered for one sentence, and text spans are mapped to the semantic elements of the corresponding frames. Each tree has four components: (1) **F**rame name (F) - the name of a frame in a pre-defined set of scenarios in FrameNet; (2) frame **T**arget (T): a lexical unit that triggers the frame; (3) frame **E**lement (E): the semantic roles of a text span of the frame; and a text span in the sentence. We represent each ⟨semantic element, text span⟩ pair as ⟨pre-terminal, terminal⟩ pairs at the leaf level, and make the frame name the root of all the pre-terminals.

The steps in construction of a SemTree are as follows: (1) extract the backbone, which is a path from the root of the frame tree to the role filler mentioning a designated object at the leaf node; (2) reverse the backbone to promote the designated object to the root; (3) attach the target and frame elements to the frame name node of the backbone. If multiple frames have been assigned to the designated object, their backbones are merged at the root node, and the frame elements (semantic roles) become siblings, as in Figure 3.

Other features

Our representation includes three vector space features: (1) bag-of-**W**ords (W; unigrams, bigrams, and trigrams); (2) word prior polarity scores from the **D**ictionary of **A**ffect in **L**anguage (DAL); and (3) bag-of-frames (FTE) for **F**rame name, frame **T**arget, and frame **E**lement.

Learning Methods

In this section we introduce tree kernel support vector machine learning for semantic trees, and an efficient kernel linearization method to create a more homogeneous representation with the hope to increase performance and to recast the learning problem from classification to regression.

Tree Kernel SVM

Support vector machine (SVM) learns a linear hyperplane $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle - b = 0$ to separate positive and negative data samples, where $\mathbf{x}$ are the samples and $\mathbf{w}$ are the weights. When taking the derivative of the Lagrangian multiplier α to plug them back to the original objective function, we have a dual problem for optimizing on α rather than $\mathbf{w}$. The optimal $\mathbf{w}$ in the original problem can be represented by a linear combination $\mathbf{w} = \sum_{i=1}^{N} y_i \alpha_i \mathbf{x}_i$. Substitution of this form into the hyperplane formula yields:

$$\begin{aligned} f(\mathbf{x}) &= (\textstyle\sum_{i=1}^{N} y_i \alpha_i \mathbf{x}_i)\mathbf{x} - b \\ &= \textstyle\sum_{i=1}^{N} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) - b = 0 \end{aligned}$$

where the inner product $\mathbf{x}_i \cdot \mathbf{x}$ extends the similarity space from Euclidean to any measure by defining $K(\mathbf{x}_i, \mathbf{x})$.

Tree representation requires a definition of the kernel function $K(T_1, T_2)$ to reflect the similarity between trees. Collins and Duffy (2001) proposed an efficient algorithm to recursively enumerate the substructure for each tree. Given two trees T_1 and T_2, $K(T_1, T_2) = \mathbf{h}(T_1) \cdot \mathbf{h}(T_2)$, function $h_i(T)$ represents the appearance of each sub-tree in the fragment space where $h_i(T) = \sum_{n \in N} I_i(n)$. $I_i(n)$ is a binary function to indicate whether the fragment i rooted n appears or not. The kernel function is hereby:

$$\begin{aligned} K(T_1, T_2) &= \mathbf{h}(T_1) \cdot \mathbf{h}(T_2) \\ &= \textstyle\sum_i h_i(T_1) h_i(T_2) \\ &= \textstyle\sum_i (\sum_{n_1 \in N_{T_1}} I_i(n_1))(\sum_{n_2 \in N_{T_2}} I_i(n_2)) \\ &= \textstyle\sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \sum_i I_i(n_i) I_i(n_2) \\ &= \textstyle\sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2) \end{aligned}$$

where $\Delta(n_1, n_2)$ is the number of common fragments rooted in the nodes n_1 and n_2. If the productions of these two nodes (themselves and their immediate children) differ, $\Delta(n_1, n_2) = 0$; otherwise iterate their children recursively to evaluate $\Delta(n_1, n_2) = \prod_j^{|children|}(1 + \Delta(c_{n_1}^j, c_{n_2}^j))$.

Linearization of the Tree Features

Linearizing the tree kernel functions converts the tree representation into a vector space by indexing tree fragments through iterative tree traversal. We applied the fragment indexing algorithm as described in (Pighin and Moschitti 2009), where sub-tree are mined and indexed efficiently. Algorithm 1 illustrates the procedure. Function $FRAG(tree)$ returns the basic fragment which consists of the root of $tree$ and its direct children. $TO\_EXPAND(frag)$ returns the set of nodes in a fragment $frag$ that can be expanded. $EXPAND(frag, maxexp)$ returns all the possible expansions of a fragment $frag$ under the limitation $maxexp$ that controls the number of $fragment$, which is the other parameter of $EXPAND$. After all the fragments are indexed, a feature vector can be produced from a tree representation by weighting each indexed feature.[3]

[3] We use raw frequency. A *idf*-adjusted value provided no improvement in a preliminary experiment using frame features.

Algorithm 1: MINE_TREE(tree)

Procedure MINE_TREE(*tree*)
$mined \leftarrow \emptyset$; $indexed \leftarrow \emptyset$; MINE_FRAG(FRAG(*tree*),*0*)
Procedure MINE_FRAG($fragment, depth$)
if $fragment \in indexed$ **then**
  Return
end
$indexed \leftarrow indexed \cup \{fragment\}$;
for $node \in TO\_EXPAND(frag)$ **do**
  if $node \in mined$ **then**
    $mined \leftarrow mined \cup \{fragment\}$;
    MINE_FRAG(FRAG($node$, 0));
  end
end
if $depth < maxdepth$ **then**
  for $fragment \in EXPAND(frag, maxexp)$ **do**
    MINE_FRAG($fragment, depth + 1$)
  end
end

Bipartite Ranking Using the Tree Features

A bipartite ranking, rather than binary classification, positions data points in a sequential order according to the probability a data instance is classified as the positive class. Data at the top of the ranked list correspond to the positive class predictions, and data at the bottom are the negative class. Using the vector space representation, we can apply two approaches to generate a bipartite ranking.

Ranking by SVM converted weights Ranking SVM is a relation-based ranking approach (Joachims 2002). The output prediction score is not a probability, but it can be used to generate a reasonable ranking.

The training procedure of ranking SVM differs from the approach discussed above. In this setting, the input training data are first ranked by their labels, that is, the positive cases rank higher than the negative cases. The objective is to learn a linear boundary $\mathbf{w}$ that $(d_1, d_2) \in Rank \Leftrightarrow \mathbf{w}d_1 > \mathbf{w}d_2$. In the training part, given training cases $\mathcal{D}$ and rank set $\mathcal{R}$, the optimization is:

$$\begin{aligned} &minimize : L(\mathbf{w}, \xi) := \tfrac{1}{2}||\mathbf{w}||^2 + C\textstyle\sum_{i,j}\xi_{i,j} \\ &s.t. \quad \forall(d_i, d_j) \in \mathcal{R}, \mathbf{w}d_i \geq \mathbf{w}d_2 + 1 - \xi_{i,j} \end{aligned}$$

where C is the slack factor for allowing training error. From the given assumption for the prediction, a rank list can be generated by sorting the testing cases by $\mathbf{w}d$.

This ranking algorithm suffers from a significant computational burden, where the number of constraints grows exponentially with the number of positive and negative data pairs. We can in fact efficiently utilize the distance from hyperplane by SVM classifier and convert the learned feature weights into a probabilistic score for ranking.

SVM outputs $f(\mathbf{x}) = \sum_{i=1}^{N} y_i\alpha_i < \mathbf{x}_i\mathbf{x} > -b$, which is an uncalibrated value that is not an probability. As an alternative, Wahba et al. (Wahba and Wahba 1998) used the logistic function to convert the general output for SVM into probabilistic form:

$$p(y_i = 1|f(\mathbf{x_i})) = \frac{1}{1 + \exp(-f(\mathbf{x_i}))}$$

where $f(x)$ is the standard output for SVM. The confidence score is hence generated.

Logistic Regression Logistic regression is another ranking algorithm we explore alongside with SVM. It assumes that the label for the given data are Bernoulli distributed. It outputs probabilities that can be considered a confidence score.

$$p(y_i|f(\mathbf{x_i})) = \frac{1}{1 + \exp(-y_i\mathbf{w}\mathbf{x_i})}$$

Log Likelihood:

$$\begin{aligned} L &:= \textstyle\sum_i \log(p(y_i|f(\mathbf{x_i}))) \\ &= -\textstyle\sum_i \log(1 + \exp(-y_i\mathbf{w}\mathbf{x_i})) \end{aligned}$$

Experiment

Our previous work combined SemTree features with vector space features and applied tree kernel learning, which resulted in an improvement over BOW features alone (Anon). Here we compare a corresponding representation with linearized versions of the SemTree features. First we test the performance on the original classification task of the original learning framework with our new linearized SemTree features. There is a statistically significant improvement in performance. We then present the ranking results. To reiterate, the feature space for both the classification task and the ranking task covers a combination of original or linearized SemTrees, bag-of-frames, including **F**rame name, **T**arget, and frame **E**lement (FTE), bag-of-words (W), and word prior polarity scores from DAL.

Classification Task

These experiments are carried out for each year, training on one year and testing on the next.[4] Two tasks are evaluated: the existence of a change and the polarity of change. We report accuracy and Matthews correlation coefficient (MCC) (Matthews 1975) for evaluation. MCC is a more robust evaluation metric that compensates the sensitivity to date skew problem for accuracy. In the time frame of our experiment, there is high variance across years in the proportion of positive labels, and often highly skewed classes in one direction or the other. MCC produces a robust summary score independent of whether the positive class is skewed to the majority or minority. In contrast to f-measure, which is a class-specific weighted average of precision and recall, and whose weighted version depends on a choice of whether the class-specific weights should come from the training or testing data, MCC is a single summary value that incorporates all 4 cells of a 2 $\times$ 2 confusion matrix (TP, FP, TN and FN for True or False Positive or Negative).

[4]In separate work, we are experimenting with a moving time window to smooth out differences across seasons and years.

$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$.

As shown in Table 1, the linearized tree kernel has higher accuracy and MCC scores for both classification tasks. The observed improvement for both evaluation metrics is statistically significant based on a one-tailed t-test. This indicates that at worst that linearization of SemTree features leads to no loss of predictive power, and at best, a gain.

Bipartite Ranking Task

One of the main reasons for linearizing the tree features is to be able to recast the problem as a ranking task. We can benefit from it to perform efficient ranking tasks in the vector space. We generate ranked lists with two methods: the probabilities derived from converted SVM weights, and logistic regression. The two methods have competitive performance, but SVM performs marginally better. Here to save space we present only the results of the semantic tree model (SemTree) using SVM weights.

We compare the performance of using all features against other combinations of features without SemTree features. Figure 4(a) illustrates the receiver operating characteristic (ROC) curves of the full ranked list from different data representations. It also presents the Area Under the ROC Curve (AUC) that correspond to each representation. As can be seen, the representation with SemTree features has higher AUC scores than the others. Its curve starts out slightly better and more stable, neck-and-neck with the other three curves at the beginning, and gradually outperforms the others all the way to the bottom of the ranked list. Figure 4(b) and 4(c) zoom in to the head and tail of the full ranked list.

The head of the ranked list is associated to the positive class (increase of price) and the tail of the list is associated to the negative class (decrease of price). The prediction at these extremes are more important than at the middle of the ranking. Table 3 provides the precision at top K for both classes. For predicting the positive label, W+FTE+DAL correctly captures 8 instances from its top 10 items, which is the best among all methods; while SemTree features starts to lead the performance after P@20. Prediction on the negative class is generally better than prediction on the positive class. In 3 out of 4 cases, SemTree features are 20% better than the second best method. We quantify the performance at the two ends of the bipartite rank list by reporting mean reciprocal rank (MRR), discounted cumulative gain (DCG), and PNorm scores (Rudin 2009). MRR and DCG are two weighted versions of AUC that favors the top (or the bottom) of the list. Higher values are preferred. PNorm score corresponds to the loss of the l_p-norms objective function, where p controls the degree of concentration to the top (or the end) of the rank list. Lower values are preferred. As can be seen in Table 3, the proposed method has better ranking performance for these three metrics.

For feature analysis, we compare the ratios of feature types by their discriminative power. As shown in Figure 5, SemTree features represent 21% of the top 1000 features ranked by information gain for polarity classification in 2010. This is representative for the other classifiers as well.

	SemTree Model	Change	Polarity
Accu.	TK	0.628 ± 0.093	0.536 ± 0.015
	LIN TK	0.636 ± 0.090	0.543 ± 0.017 *
MCC	TK	0.130 ± 0.032	0.073 ± 0.030
	LIN TK	0.171 ± 0.043 **	0.088 ± 0.033 **

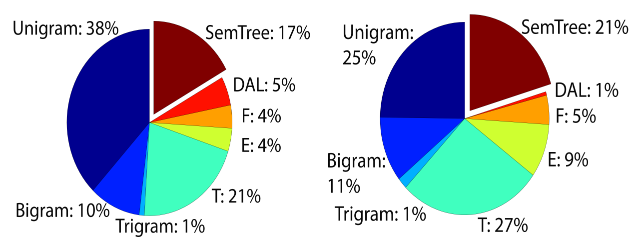
Table 1: Classification results for our experimental time frame 2007-2012. Mean and standard deviation of the accuracy and MCC scores are reported. * indicates a $p-value < 0.1$ for the one-tailed t-test; ** indicates a $p-value < 0.05$.

Data Representation	P@10	P@20	P@50	P@100
positive class (increase of price)				
W+DAL	0.7	0.5	0.52	0.46
FTE+DAL	0.6	0.45	0.38	0.45
W+FTE+DAL	0.8	0.45	0.44	0.46
SemTree+W+FTE+DAL	0.5	0.5	0.54	0.55
negative class (decrease of price)				
W+DAL	0.6	0.55	0.54	0.46
FTE+DAL	0.6	0.75	0.54	0.49
W+FTE+DAL	0.8	0.6	0.54	0.51
SemTree+W+FTE+DAL	1	0.75	0.68	0.63

Table 2: Precision@Top K evaluation for positive and negative class predictions.

Data Representation	MRR	DCG	PNorm64
positive class (increase of price)			
W+DAL	0.354	298.167	7.31E+220
FTE+DAL	0.355	298.245	7.87E+220
W+FTE+DAL	0.354	298.189	7.90E+220
SemTree+W+FTE+DAL	0.357	298.414	6.46E+220
negative class (decrease of price)			
W+DAL	0.350	294.502	3.14E+220
FTE+DAL	0.351	294.594	2.87E+220
W+FTE+DAL	0.351	294.530	3.08E+220
SemTree+W+FTE+DAL	0.353	294.777	1.87E+220

Table 3: Evaluation that concentrates on positive and negative predictions by DCG, MRR and PNorm (lower is better).

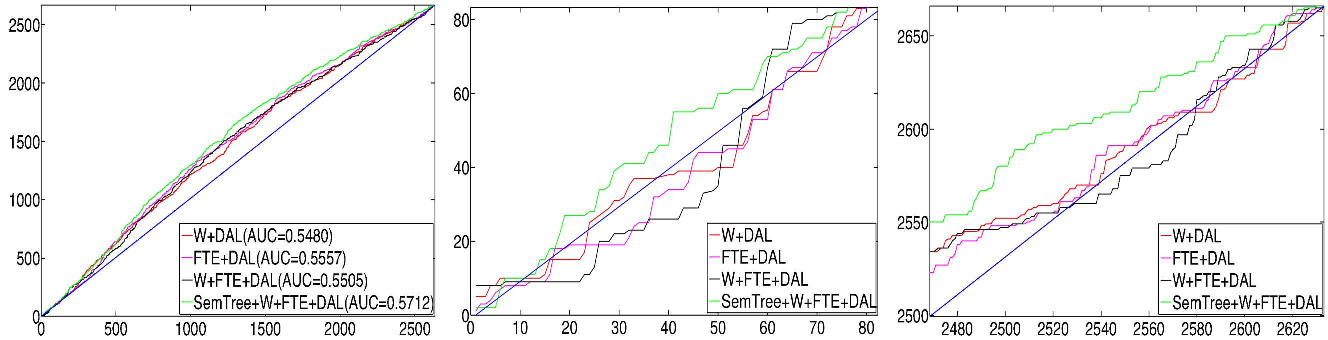


(a) top 100 (b) top 1000

Figure 5: Ratio of feature types at top 100 and top 1000 ranked list by information gain for 2010 polarity prediction.

Conclusion

This study compares alternative feature representations to capture news impact on the market. The results demonstrate that linearization of the SemTree features we developed to



(a) Raw SVM: Full ROC (b) Raw SVM: Head ROC (c) Raw SVM: Tail ROC

Figure 4: ROC curves for the polarity task

capture general semantic frame dependencies of companies in the news leads to superior classification results. It also supports the application of a wide range of ranking approaches that require vector space feature representation. On the classification tasks to predict change and direction of stock price, this achieves a more efficient SVM computation than in our previous work (Anon). We also rank data instances according to their probability to affect the change of price by SVM converted scores and logistic regression. We evaluate our rankings with ROC curves, and metrics that quantify the contribution at both ends of a ranked list.

Future work will consider contextual information for sentence selection, and an aggregation of weighted news content that accounts for decay over time. We plan to use a moving window for training and testing. We will also explore different labeling methods, such as a threshold for price change tuned by sectors and background economics.

References

Baker, C. F.; Fillmore, C. J.; and Lowe, J. B. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the ACL and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, 86–90. Stroudsburg, PA, USA: ACL.

Collins, M., and Duffy, N. 2001. Convolution kernels for natural language. In *Proceedings of NIPS2001*.

Das, D., and Smith, N. A. 2012. Graph-based lexicon expansion with sparsity-inducing penalties. In *HLT-NAACL*, 677–687. The ACL.

Feldman, R.; Rosenfeld, B.; Bar-Haim, R.; and Fresko, M. 2011. The stock sonar - sentiment analysis of stocks based on a hybrid approach. In *Proceedings of the Twenty-Third Conference on Innovative Applications of Artificial Intelligence, August 9-11, 2011, San Francisco, California, USA*.

Fillmore, C. J. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences* 280(1):20–32.

Joachims, T. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 133–142. ACM.

Kogan, S.; Levin, D.; Routledge, B. R.; Sagi, J. S.; and Smith, N. A. 2009. Predicting risk from financial reports with regression. In *Proceedings of HLT: The 2009 Annual Conference of the North American Chapter of the ACL*, NAACL '09, 272–280. Stroudsburg, PA, USA: ACL.

Lavrenko, V.; Schmill, M.; Lawrie, D.; Ogilvie, P.; Jensen, D.; and Allan, J. 2000. Mining of concurrent text and time series. In *In proceedings of the 6th ACM SIGKDD Int'l Conference on KDD Workshop on Text Mining*, 37–44.

Luss, R., and d'Aspremont, A. 2011. Predicting abnormal returns from news using text classification.

Matthews, B. W. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405(2).

Pighin, D., and Moschitti, A. 2009. Reverse engineering of tree kernel feature spaces. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore*.

Rudin, C. 2009. The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list. *J. Mach. Learn. Res.* 10:2233–2271.

Schumaker, R. P., and Chen, H. 2010. A discrete stock price prediction engine based on financial news. *IEEE Computer* 43(1):51–56.

Schumaker, R. P.; Zhang, Y.; Huang, C.; and Chen, H. 2012. Evaluating sentiment in financial news articles. *Decision Support Systems* 53(3):458–464.

Tetlock, P. C.; Saar-Tsechansky, M.; and Macskassy, S. 2008. More than Words: Quantifying Language to Measure Firms' Fundamentals. *The Journal of Finance*.

Tetlock, P. C. 2007. Giving Content to Investor Sentiment: The Role of Media in the Stock Market. *The Journal of Finance*.

Wahba, G., and Wahba, G. 1998. Support Vector Machines, Reproducing Kernel Hilbert Spaces and the Randomized GACV.

Wang, S.; Xu, K.; Liu, L.; Fang, B.; Liao, S.; and Wang, H. 2011. An ontology based framework for mining dependence relationships between news and financial instruments. *Expert Systems with Applications* 38(10):12044 – 12050.