Semantic Feature Representation to Capture News Impact

Boyi Xie and **Dingquan Wang** and **Rebecca J. Passonneau**
Center for Computational Learning Systems
Columbia University
New York, New York 10115
{xie@cs.|dw2546@|becky@ccls.}columbia.edu

Abstract

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 machines 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 learning tasks, 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 real world observations. Recent work has applied a variety of automated methods to news, from topic modeling to more linguistically informed techniques. For example, if a company becomes the target of a lawsuit, this could affect investors' perception of the company's value. To take such facts into account, however, it is necessary to distinguish the participants: which company is the plaintiff and which one is the defendant. A conventional bag-of-words (BOW) model cannot capture this information. Our previous work (Xie et al. 2013) 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, 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 et al. (2009) analyzed quarterly earning reports to predict stock return volatility and to predict whether a company will be delisted. Luss and d'Aspremont (2011) used text classification to model price movements of financial assets on a per-day basis. They tried to predict the direction of return and abnormal returns, defined as an absolute return greater than a predefined thresh-

old. Schumaker and Chen (2010) proposed a stock price prediction system based on financial news. They represent documents by boolean valued BOW and named entities. Wang et al. (2011) presented a framework for mining the relationships between news articles and financial instruments using a rule-based expert system. 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 BOW model. In this study, we explore a rich feature space for 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. (2012) treat stock price prediction as a sentiment analysis problem to distinguish positive and negative financial news. Tetlock, Saar-Tsechansky, and Macskassy (2007; 2008) quantify pessimism of news using General Inquirer (GI), a content analysis program. Feldman et al. (2011) applies sentiment analysis on financial news using rule-based information extraction and dictionary-based prior polarity scores. 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.

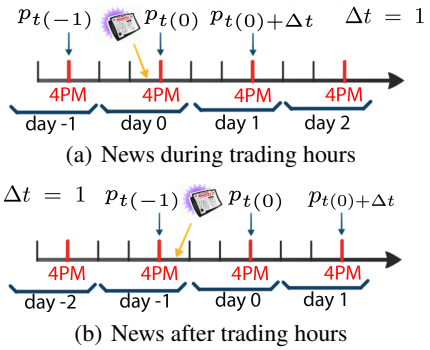
Data and Labels

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 consumer staples, which is a medium size sector in the GICS.[2] 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 $\Delta t = 1$ (day) 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 company or time.



(a) News during trading hours

(b) News after trading hours

Figure 1: Aligning stock price with news.

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

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

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 with a price change are included in the *polarity* task.

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, thus a SemTree concisely represents all the semantic roles for a given company mentioned in one sentence.

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[3] (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 *Colgate sued GlaxoSmithKline*, where Judgment_communication is the frame (F) triggered by target (T)

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

[2] Global Industry Classification Standard.

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

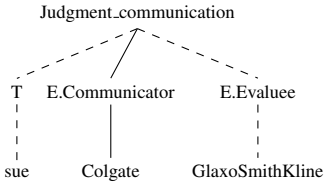


Figure 2: Frame tree for *Colgate sued GlaxoSmithKline*.

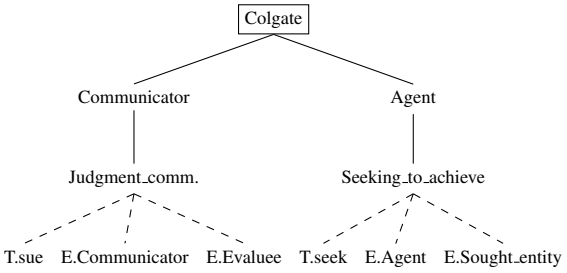


Figure 3: Constructing a SemTree for *Colgate* from the sentence *Colgate sued GlaxoSmithKline, seeking a court order that its packaging for Colgate toothpaste does not infringe trademarks held by the maker of Aquafresh.*

sue, with text span *Colgate* being the COMMUNICATOR element, and *GlaxoSmithKline* being the EVALUEE element.

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 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 (4) a text span in the sentence.

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. The goals were to improve performance, and to recast the learning problem from classification to regression.

Tree Kernel SVM

Support vector machines (SVMs) learn 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 a variant of the fragment indexing algorithm described in (Pighin and Moschitti 2009). Algorithm 1 illustrates the procedure. It resembles a depth-first search procedure that starts with the root and generates one-level (a node and its direct children) fragment features by traversing the tree all the way down to the pre-terminals that produce leaves. When it traces back from the pre-terminals, multi-level fragment features are also indexed. Because the depth of our trees is three, the sub-tree fragment features are mined and indexed efficiently. Function $PROD(n)$ returns the production of n, i.e. one-level fragment which consists of node n and its direct children. The $mined$ set is used so that the base fragment rooted in a given node is considered only once. After all the fragments are indexed, a feature vector can be produced from a tree representation by weighting each indexed fragment.[4]

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

Algorithm 1: Linearize a Tree to Vector Space Features

```
Procedure LINEARIZE_TREE(tree)
features ← ∅; mined ← ∅; MINE_FRAG(PROD(tree))

Procedure MINE_FRAG(frag)
if frag ∈ features then
    Return
end
features ← features ∪ {frag};
for node ∈ {nodes in frag and are internal nodes of the
original tree} do
    if node ∉ mined then
        mined ← mined ∪ {node};
        MINE_FRAG(PROD(node));
    end
end
for f ∈ {fragments of all possible expansions created by
including direct children of a enumeration of frag's nodes}
do
    MINE_FRAG(f);
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.

In this setting, the training data are first ranked by their labels, i.e. the positive cases rank higher than the negative cases. Given training cases $\mathbf{x} \in \mathcal{T}$ and rank set $\mathcal{R}$, where $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{R}$ if $\mathbf{x}_i$ ranks higher than $\mathbf{x}_j$. We want to learn the weights $\mathbf{w}$ so that the maximum number of pairs $(\mathbf{x}_i, \mathbf{x}_j)$ satisfy $\mathbf{w} \cdot \mathbf{x}_i > \mathbf{w} \cdot \mathbf{x}_j$. By introducing (non-negative) slack variables $\xi_{i,j}$, the optimization problem becomes

$$minimize : L(\mathbf{w}, \xi) := \tfrac{1}{2}||\mathbf{w}||^2 + C \sum_{i,j} \xi_{i,j}$$
$$s.t. \qquad \forall (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{R}, \mathbf{w} \cdot \mathbf{x}_i \geq \mathbf{w} \cdot \mathbf{x}_j + 1 - \xi_{i,j}$$

where C is a parameter that allows trading-off margin size against training error. A rank list can be generated by sorting the testing cases by $\mathbf{w} \cdot \mathbf{x}$.

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 $f(\mathbf{x}) = \sum_{i=1}^{N} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) - b$, which is an uncalibrated value that is not an probability. As an alternative, 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.

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

where $\mathbf{w}$ is estimated by maximizing the log of the conditional likelihood:

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

where $y_i = \{1, -1\}$. For testing cases $\mathbf{x}$, $p(y = 1 | f(\mathbf{x}))$ is used for bipartite ranking.

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 (Xie et al. 2013). 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.[5] 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 2×2 confusion matrix (TP, FP, TN and FN for True or False Positive or Negative).

[5] 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 MCC evaluation metric 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 the rankings by SVM converted 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 2 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 ranked 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 ranked 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 **

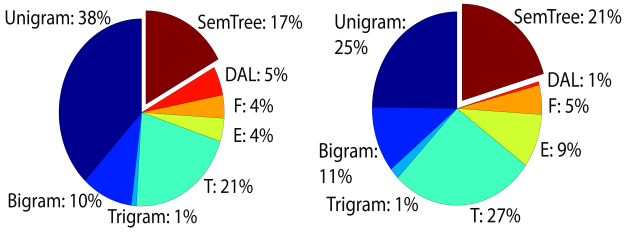
Table 1: Classification results for our experimental time frame 2007-2012. Mean and standard deviation of the accuracy and MCC scores are reported. * indicates a $p-value < 0.1$ for the one-tailed t-test; ** indicates a $p-value < 0.05$.

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 a total of N=5290 instances in year 2010 for pos and neg 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).

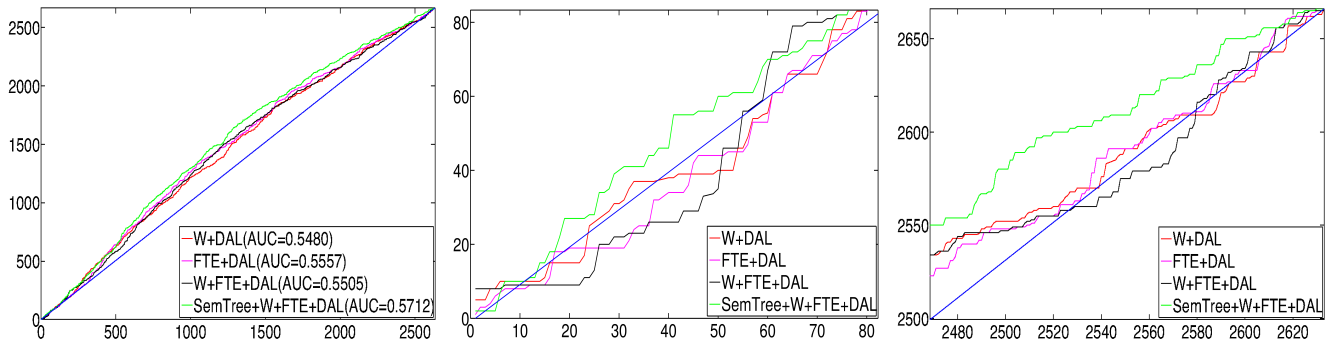


(a) top 100 (b) top 1000

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 to capture general semantic frame dependencies of companies in the news



(a) Full ROC curve (b) Head ROC curve (c) Tail ROC curve

Figure 4: Receiver operating characteristic (ROC) curve for the polarity task.

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 (Xie et al. 2013). 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.

For future work, we will explore more feature engineering and data representations that consider contextual information among sentences in news documents, and efficient kernel learning methods. We here provide an exemplary approach for fine-grained (target-oriented) sentiment analysis with semantic features. It can be applied to datasets in other domains, such as political news or product reviews.

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.

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.

Xie, B.; Passonneau, R. J.; Wu, L.; and Creamer, G. 2013. Semantic frames to predict stock price movement. In *ACL*, 873–883.