Connecting The Dots: Forecasting and Explaining Short-Term Market Volatility

Jie Yuan**
Iowa State University
Ames, USA
jieyuan@iastate.edu

Zhu Zhang*
Iowa State University
Ames, USA
zhuzhang@iastate.edu

ABSTRACT

Market volatility prediction is of significant theoretical and practical importance in the financial market, and the news is a significant source to influence the market. By using deep learning networks, we can forecast the volatility based on the news; meanwhile, how to explain the deep neural network is a prevalent topic, especially the attention mechanism in the NLP field. Current studies mainly focus on unveiling the principles behind attention mechanisms without considering generating human-readable explanations. In this work, we attempt to generate a human-readable explanation about the evidence that led to the prediction. To achieve our goal, we propose news-powered neural models to forecast short-term volatility and present a soft-constrained dynamic beam allocation algorithm to control the state-of-the-art language model (GPT-2) to generate fluent and informative explanations.

CCS CONCEPTS

• **Applied computing** → **Economics**; • **Computing methodologies** → *Natural language generation*; *Neural networks*.

KEYWORDS

market volatility, neural networks, forecasting, attention mechanisms, explanation

ACM Reference Format:

Jie Yuan and Zhu Zhang. 2020. Connecting The Dots: Forecasting and Explaining Short-Term Market Volatility. In *ACM International Conference on AI in Finance (ICAIF '20)*, October 15–16, 2020, New York, NY, USA. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3383455.3422518>

1 INTRODUCTION

Market volatility (MV) reflects the degree of variability in prices within a fixed time interval, such as in a day, a month, or a year. More specifically, it is the amount of risk related to the size of changes in a market's value. A higher volatility of the market indicates a sudden price change in either direction. Due to its uncertain characteristic, MV can have a significant impact on the decision of any investor. Hence, predicting market volatility becomes a critical issue in financial market research and practice.

*Both authors contributed equally to this research.

This work is licensed under a Creative Commons Attribution International 4.0 License.
ICAIF '20, October 15–16, 2020, New York, NY, USA
© 2020 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7584-9/20/10.
<https://doi.org/10.1145/3383455.3422518>

Figure 1: The workflow of this work (the square in vertical vector denotes the word attention score; the square in the horizontal vector represents the sentence attention score).

In order to forecast MV, dozens of variables contribute and interact with each other. However, in this work, we focus our efforts on utilizing unstructured text data as the main ingredient. Specially, news titles [7] carry valuable information which we leverage to indicate volatility about a market index (e.g. S&P500). Furthermore, currently, many state-of-the-art (SOTA) models are influenced by deep neural networks (DNNs) for MV prediction. Serious efforts such as Neural Stochastic Volatility Model (NSVM) [16] and Stock-Net [23] utilize historical prices and social media tweets to make the prediction, yet they miss the opportunity to exploit news titles about the market index. Therefore, we build a news-powered attention based neural models to forecast the short-term MV.

Apart from improving the performance of DNNs, attention mechanism has become the *de facto* instrument for model inspection, where one can gain shallow insights about which features are relevant for a particular prediction. Specific to our problem, the attention mechanism may allow us to identify which tokens in news titles contribute towards the prediction of MV. However, instead of just listing the tokens with highest attention, we push the envelope further. Inspired by [1], in this effort, we attempt to generate human-readable explanations that led to the MV prediction by exploiting state-of-the-art language models such as GPT-2 [19].

Furthermore, very limited focus has been devoted towards modification of GPT-2 by applying search algorithm [11]. [10, 12, 17] propose constrained beam search algorithms, which extend vanilla beam search to allow the inclusion of pre-specified lexical constraints and perform well on machine translation (MT) task. However, one common issue in these constrained beam search algorithms is that they are applicable on a sequence to sequence model. Whereas, GPT-2 is similar to a decoder-only transformer and does not contain the encoder. According to us, applying the mentioned constrained beam search algorithms directly on GPT-2 will likely output sentences without containing any pre-specified lexical constraints. Hence, we propose a **Soft Constrained Dynamic Beam Allocation (SC-DBA)** algorithm, which extends the lexical constraints and aims to cover constraint token as much as possible.

In summary, our contributions are as below:

- (1) To utilize news titles about market index, we create a novel news dataset from the Wall Street Journal (WSJ).
- (2) To solve the data sparsity in our news dataset, we introduce a unique data augmentation technique.
- (3) To forecast short-term volatility, we present several attention-based news-powered neural models.
- (4) To generate a human-readable explanation and for model-interpretability, we propose SC-DBA algorithm which can control the generation of GPT-2 model

2 RELATED WORK

One way to forecast market volatility is based on the historical data [2, 8, 15]. Recently, [16] introduced a neural stochastic volatility model (NSVM) to predict MV. It is a stochastic recurrent neural network framework that can formulate the temporal dynamics of volatility. In contrast to our proposed models, NSVM predicts MV based on target variables without incorporating public news.

Another way to predict volatility is by incorporating natural language resources such as news titles. [7] extracts events from news text, models the influence of long and short term events by using conventional neural networks, and predicts the price movements. [6] forecasts the price movement by adopting the Open IE technology, whereas, [14] designs a hybrid attention network to predict the price trend. Most recently, [23] proposes a model, StockNet, to predict stock movement from tweets and historical prices by introducing a recurrent and continuous latent variable to better simulate stochastic processes.

Data imbalance is a well-studied technique in classification setting where it is conventional to employ oversampling on the class which has fewer samples. However, for regression, the oversampling approach is not applicable. Only a few pre-processing methods

are available to deal with imbalanced regression problems. [20] develops a method, SmoteR, to address the issues of predicting rare extreme values of a continuous target variable. The method can generate synthetic new features and target variables based on the K nearest neighborhoods of the seeds. Moreover, [3] proposes a Gaussian Noise method which introduces a small perturbation in the features and target variable, and the noise depends on a fraction of the standard deviation of each numeric feature. [4] combines the ideas of the SmoteR and Gaussian Noise by introducing safe distance and proposes SMOGN.

In addition to predicting MV, we aim to generate a human-readable explanation about the evidence that led to the prediction [1]. [21] proposes a novel, simple yet strong network architecture, Transformer, which is based solely on the attention mechanism. Furthermore, following models: BERT [5] and GPT-2 [19] have been developed based on the Transformer model. BERT is a powerful encoder that can capture long contextual information and GPT-2, a successor of GPT [18], is an impressive generator which can generate quality texts. In this paper, we utilize BERT to encode our news titles and employ GPT-2 to generate the explanations.

Since generation is a search problem virtually, several constrained beam search algorithms have been proposed to generate text containing pre-specified constraints. [10] propose a grid beam search (GBS) algorithm, which constrains the beam search to contain constraints words, and has been applied on machine translation (MT) problem. The GBS can guarantee that generated sentences cover all constraints; however, it's too inefficient to be applied at a large scale. Inspired by [10], [17] propose a dynamic beam allocation (DBA) algorithm, which extends the candidate hypotheses to generate lexically constrained hypotheses. Then, [12] propose a vectorized dynamic beam allocation (VDBA) algorithm, a successor of DBA. VDBA improves the DBA by using a vectorized approach to allocate beam and using tries to record constraint state. Different from VDBA, since our lexical constraints are temporal sequences, we don't need tries to avoid corner cases of phrases.

3 PROBLEM FORMULATION

As shown in Figure 1, firstly, we formulate the MV prediction as a regression task and predict the volatility based on the given news title. Next, we classify instances, a set of tuple (news, true value, prediction), into three classes: high, medium and low volatility. Then, we extract the seed words (e.g. *credit*, *real state*, *fund*) by utilizing attention scores, form the constraints (e.g. *[[credit], [real state], [fund]]*), and generate the explanation (e.g. *The credit card company for the fund*) about the evidence (words) which lead to the prediction.

For MV prediction, we aim at predicting the volatility, SPV_{t1} , of S&P 500 (SP) index when the stock market closes at time t on a target trading day d . The volatility (V_t) is defined as the difference between anchor day's high and low price divided by the average of anchor day's open and close prices. The SPV_{t1} is defined as follow:

$$SPV_{t1} = \frac{Max(PriceHigh_{t:t+1}) - Min(PriceLow_{t:t+1})}{Avg(PriceClose_{t-1}, PriceClose_{t+1})} \quad (1)$$

In our case, the problem of MV prediction can be formalized as: given a target trading day d , a stock market closes at time t_d on d and a time window W_d is defined, where $W_d = [t_d - \Delta h, t_d -$

Models (SPVt1)	MAPE	MAE	MSE	Precision	Recall	F1-value
NSVM	129.6701	0.010483	0.000146	0.132	0.026	0.044
StockNet	180.1796	0.016394	0.000392	0.059	0.323	0.095
Attention-LSTM	103.2686	0.009182	0.000136	0.109	0.041	0.036
Attention-LSTM-Attention	113.9557	0.009679	0.000142	0.041	0.088	0.045
Attention-Attention	107.5648	0.009051	0.000124	0.132	0.085	0.096
Nested-Attention-LSTM	110.2148	0.009649	0.000147	0.191	0.111	0.093
Nested-Attention-LSTM (Aug)	135.5256	0.011440	0.000194	0.201	0.373	0.152
Mean baseline	131.3322	0.010584	0.000148	0.000	0.000	0.000

Table 1: Performance of baselines and our models to predict SPVt1 on test dataset.

1_{hour}] and Δh is a fixed lag size. We first split the time window in K time segments uniformly where the length of each time segments is $(\Delta h - 1)/K$ hours. Then, W_d can be denoted as: $W_d = [seg_1, seg_2, \dots, seg_K]$. In each time segment $seg_k, k \in [1, K]$, there are N news titles NT , $seg_k = [NT_1, NT_2, \dots, NT_N]$, and all news titles in seg_k are sorted in accordance with Unix time¹. For each news title $NT_n, n \in [1, N]$, it contains L tokens TK , $NT_n = [TK_1, TK_2, \dots, TK_L]$. Our goal is to use the summarized news titles sequence in the time window W_d to predict $SPVt1$.

The problem of generating explanations can be formulated as an instance of lexically-constrained decoding which is a variation of beam search that forces the output to contain pre-specified words and phrases.

4 MODELS FOR VOLATILITY PREDICTION

In this section, we present the framework of our models, which consist of news title encoder, sequential modeling, and volatility prediction. Next, to solve the data sparsity, we propose a new data augmentation technique.

4.1 Model Components

News Titles Encoder: To reduce the complexity of our models, we take advantage of pre-trained BERT² [5] [22] to tokenize, and encode a news title NT_n to a sequence of token embedding $NT_n^E = [TK_1^E, TK_2^E, \dots, TK_L^E]$, where $TK_l^E \in \mathbb{R}^{1 \times d_m}, l \in [1, L]$. We perform two methods to construct news title embedding: (1) We feed news title into pre-trained BERT, extract the last second hidden layer, and average the extracted layer to construct a news title vector $NT_n^{Ave} \in \mathbb{R}^{1 \times d_m}$. (2) Since not all the words in the titles are useful, we use a word-level self-attention to filter words and then gather the token embeddings of words weighted by the assigned attention score. Specifically, word-level self-attention layer is applied as below:

$$w_{NT} = \zeta(\mathbf{W}_{att}^T ReLu(\mathbf{W}_e NT_n^E)) \quad (2)$$

where $\zeta(\cdot)$ is softmax function and $\mathbf{W}_{att} \in \mathbb{R}^{d_m \times 1}, \mathbf{W}_e \in \mathbb{R}^{d_m \times d_m}$ are model parameters. Then, we compute a weighted sum over NT_n^E as below.

$$NT_n^{Att} = \sum w_{NT} NT_n^E \quad (3)$$

Therefore, the news titles encoder can provide two news title embeddings, NT_n^{Ave} or NT_n^{Att} . Finally, seg_k can be represented as

$seg_k^E = [NT_1^{Ave}, NT_2^{Ave}, \dots, NT_N^{Ave}]$ based on NT_n^{Ave} or $seg_k^E = [NT_1^{Att}, NT_2^{Att}, \dots, NT_N^{Att}]$ based on NT_n^{Att} .

Sentence-level self-attention: As not all the news titles make even contribution while predicting volatility, we calculate a weighted sum of news title embeddings in time segment seg_k ,

$$w_{seg} = \zeta(\mathbf{W}_{atts}^T ReLu(\mathbf{W}_s seg_k^E)) \quad (4)$$

where $\zeta(\cdot)$ is softmax function and $\mathbf{W}_{atts} \in \mathbb{R}^{d_m \times 1}, \mathbf{W}_s \in \mathbb{R}^{d_m \times d_m}$ are model parameters. Next, we compute a weighted sum over seg_k^E .

$$seg_k^{Att} = \sum w_{seg} seg_k^E \quad (5)$$

After obtaining seg_k^{Att} , time window W_d can be represented as $W_d^E = [seg_1^{Att}, seg_2^{Att}, \dots, seg_K^{Att}]$.

Sequential Modeling: At this point, we have a temporal sequence of summarized news title embedding W_d^E . To capture the temporal information, we use long short term memory (LSTM) network [9] to encode W_d^E , and obtain a hidden state h_k for step k .

$$h_k = LSTM(seg_k^{Att}, h_{k-1}), k \in 1 \dots K \quad (6)$$

Volatility Prediction: The last component of our models is a fully connected layer and linear regression layer, which takes the last hidden layer h_K of LSTM as an input and predicts the volatility y_d at market close time t_d on the trading day d .

$$\hat{y}_d = \mathbf{W}_1(\mathbf{W}_f h_K + \mathbf{b}_f) + \mathbf{b}_1 \quad (7)$$

where $\mathbf{W}_1 \in \mathbb{R}^{1 \times d_h}, \mathbf{W}_f \in \mathbb{R}^{d_h \times d_h}, \mathbf{b}_1 \in \mathbb{R}^1$ and $\mathbf{b}_f \in \mathbb{R}^{d_h}$ are model parameters, d_h is the dimension of hidden states.

Now, we combine our model components in different ways and propose following four models: (a) *Attention-LSTM* is a stack of the news titles encoder (1), the sentence-level self-attention, the sequential modeling and the volatility prediction; (b) *Attention-LSTM-Attention* is a stack of the news titles encoder (1), the sentence-level self-attention, the sequential modeling, sentence-level self-attention, and the volatility prediction. The input of the second sentence-level self-attention are all hidden states $[h_1, h_2, \dots, h_K]$; (c) *Attention-Attention* is a stack of the news titles encoder (1), two sentence-level self-attention layers and the volatility prediction, where the input of the second sentence-level self-attention layer is seg_k^{Att} (d) *Nested-Attention-LSTM* is a stack of the news titles encoder (2), the sentence-level self-attention, the sequential modeling and the volatility prediction.

In essence, all of our various models digest news titles as inputs to forecast the short-term volatility, MV.

¹Unix time is the number of seconds that have elapsed since the Unix epoch, minus leap seconds; the Unix epoch is 00:00:00 UTC on 1 January 1970.

²https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-uncased-pytorch_model.bin

4.2 Data Augmentation

Post creation of our dataset, we noticed that the dataset is sparse where true values in training and validation datasets follow a skewed distribution. In previous works [20] [3] [4], it was reasonable to generate synthetic data by interpolating features to solve the sparsity issue. However, since our input consists of word embeddings or sentence embeddings, it is hard to explain the meaning of interpolation of two embeddings. To overcome this issue, we rewrite the original input sentences and then use BERT to encode the paraphrases of input sentences. [13] introduced the ParaBank³, a large-scale English paraphrase dataset, and trained a machine translation (MT) model on the English paraphrase dataset. By leveraging the pre-trained MT model, we can rewrite the original sentences. **Data augmenting range:** Since the financial market cares more about high values, we consider the true values which are greater than the mean as high values. As training and validation datasets follow a skewed distribution, a large number of true values are in the range $[\mu, \mu + \sigma)$, where μ and σ are mean and standard deviation of the training dataset. If we over-sample instances from the range $[\mu, \text{inf})$ uniformly, most of the sampled instances will come from the range of $[\mu, \mu + \sigma)$ and the spikes (true values in $[\mu + \sigma, \text{inf})$) will not be sampled. To alleviate this issue, we sample the instances whose true value lies in the ranges $[\mu, \mu + \sigma)$ and $[\mu + \sigma, \text{inf})$ uniformly. **Augmenting data:** Formally, we sample m instances from $[\mu, \mu + \sigma)$ and n instances from $[\mu + \sigma, \text{inf})$ separately, where m and n are hyper-parameters. For each instance, we use Algorithm 1 to generate k new instances by repeating Algorithm 1 k times. Finally, there are $k(m + n)$ new over-sampled instances.

Algorithm 1: Data Augmentation Algorithm

Input: (sentences S , target variable Y) in instance
Parameters: number of paraphrases N
 new_sentences = [];
foreach s **in** S **do**
 paraphrases = []
 para_candidates = Rewriter(s , N);
 foreach p **in** $para_candidates$ **do**
 if $0.98 \leq \text{cosineSimilarity}(s, p) \leq 0.99$ **then**
 Paraphrases.append(p)
 else
 continue;
 if $paraphrase$ **is not empty** **then**
 $s = \text{sample}(paraphrase, 1)$;
 new_sentences.append(s);
Output: (new_sentences, Y)

5 ALGORITHM FOR TEXT GENERATION

In this section, we describe the process of creating positive constraints and the algorithm SC-DBA for generating explanations.

Classify Instances: Since our model has a word-level attention mechanism, we can infer which words contribute more to the MV prediction and generate the explanation. However, since $SPVt1$ is a continuous variable, it is hard to see the correlation between

the explanation and the prediction. Therefore, we discretize the volatility and classify I instances⁴ into different classes,

$$set_I = \{(W^1, y^1, \hat{y}^1)_1, (W^2, y^2, \hat{y}^2)_2, \dots, (W^I, y^I, \hat{y}^I)_I\}$$

where W^i represents all the titles in a time window, y^i is the true value, and $\hat{y}^i$ is the prediction for the instance i , $i \in [1, I]$. Given a set of instances set_I , we can obtain a true value set $Y = \{y^i | y^i \in set_I\}$. Then, we define high/low threshold as λ_h/λ_l based on Y where $\lambda_l = \mu$, $\lambda_h = \mu + \sigma$, μ is the mean and σ is the standard deviation of Y . According to high/low threshold, we can classify instances into three classes, $C_{\geq \lambda_h}$, $C_{[\lambda_l, \lambda_h)}$ and $C_{< \lambda_l}$.

$$Classes = \begin{cases} C_{\geq \lambda_h}, & \text{if } y, \hat{y} \geq \lambda_h \\ C_{[\lambda_l, \lambda_h)}, & \text{if } \lambda_l \leq y, \hat{y} < \lambda_h \\ C_{< \lambda_l}, & \text{if } y, \hat{y} < \lambda_l \end{cases} \quad (8)$$

Extract Seed Words: After classifying instances into different classes, we extract seed words from W^i (equivalent to W_d) by taking advantage of sentence-level self-attention scores w_{seg} in equation (4) and word-level self-attention scores w_{NT} in equation (2). Next, we define a sentence-level threshold γ_{sent} to filter titles and a word-level threshold γ_{word} to filter words. In Algorithm 2, we describe our process to extract seed words. Specifically, the *mergeToken* method corresponds to the condition: if the $Token_l$ is in a phrase with two tokens (*corresponds to* $Token_l$ is continues), $[Token_{l-1}]$ $[Token_l]$ will be merged into $[Token_{l-1}, Token_l]$ in the seeds.

Algorithm 2: Extract Seed Words

Input: W_d , w_{NTn} , $n \in [1, \dots, N]$, w_{segk} , $k \in [1, \dots, K]$
Parameters: number of segments K , titles N , words L
 seeds = [];
for k **in** $\text{range}(K)$ **do**
 for n **in** $\text{range}(N)$ **do**
 if $w_{segk}[n] > \gamma_{sent}$ **then**
 for l **in** $\text{range}(L)$ **do**
 if $w_{NTn}[l] > \gamma_{word}$ **then**
 if $Token_l$ **is continues** **then**
 mergeToken(seeds, $Token_l$);
 else
 seeds.append($Token_l$);
 else
 continue;
 else
 continue;
Output: seeds

Finally, we obtain the seed words list. Since some seed words are word pieces, by pre-processing seed words list, we get final positive constraints, which refer to the tokens or token sequence which must be present in the generated sentence (in contrast, negative constraints refer to the tokens or token sequence which must not be generated) [12], denoted as

$$C = [[w_1], [w_2, w_3], [w_4 w_5, w_6], [w_7], \dots, [w_S]]$$

where w_i , $i \in [1, S]$ represents one constraint token, $[w_2, w_3]$ represents w_2 and w_3 are extracted from a same title, $w_4 w_5$ in $[w_4 w_5, w_6]$

³<http://decomp.io/projects/parabank/>

⁴The validation dataset or the testing dataset can be considered as set_I .

is a phrase. Given positive constraints, our next step is to generate an explanation by using GPT-2 [19]. [17] propose a dynamic beam allocation (DBA) algorithm, which extends candidate hypotheses of traditional beam search following three rules: (1) the best k tokens across all the row of Scores. (2) add all unmet constraints for current hypotheses. (3) the best token for the current hypothesis in the beam. However, the DBA algorithm cannot be directly applied on GPT-2 because it works on the sequence to sequence models having encoder-decoder. Since GPT-2 is only a transformer-based decoder, it is possible to generate a text without containing any positive constraint tokens.

In order to generate a fluent, informative explanation (sentences) and guarantee the presence of constraint tokens, we improved the dynamic beam allocation (DBA) algorithm as follows: (1) expand the single constraint token by adding the *top-k* most similar (based on cosine similarity) tokens to the single constraint token; (2) utilize a Boolean array to record if constraints are covered, we use similarity score to record the status of constraints and also to weigh the score of candidate hypothesis; (3) modify the objective function to weighted objective function, so that the outputs will contain as many constraints as possible and at the same time retain the semantic information. Therefore, we propose an improved soft constrained dynamic beam allocation algorithm (SC-DBA) (Algorithm 3). The details of the Algorithm 3 are as follows:

Extend constraints: The motivation of extending constraints is that the seed words may have different senses. By extending constraints, it is possible to generate a more fluent and natural sentence which preserves the original semantic information. In practice, we simplify the implementation of constraints C and convert the constraints C to $C' = [[w_1], [w_2], [w_3], [w_4, w_5], [w_6], [w_7], \dots, [w_S]]$.

To extend the constraints, we only extend single constraint token, such as w_1, w_2, w_3, w_6, w_7 and w_S , by appending *top-k* most similar words $w_1^1 \dots w_i^k$ based on word similarity. The extended w_i can be denoted as $w_i : [w_i^1 \dots w_i^k], i \in [1, 2, 3, 6, 7, S]$. (Pseudo-code refers to the first for loop in the Algorithm 3.)

Coverage array: In order to track how many constraints are covered in a hypothesis, we use a *coverage array* to record the number and status of the constraint tokens in a hypothesis. The length of *coverage array* is same as C' . *Coverage array* is denoted as

$$coverArray = [[c_1], [c_2], [c_3], [c_4], [c_5], [c_6], \dots, [c_S]]$$

where $c_j \in [0, 1], j \in [1, S]$. Different from [19][17] using binary array ($c_j = 0$ or 1), the c_j is defined by computing the cosine similarity of two words embedding (constraint token and one token in *top-k*). So that each cell in *coverage array* not only reflects the status of constraint token but also the semantic similarity between the constraint token w_i and an extended constraint token in $[w_i^1 \dots w_i^k]$.

Weighted objective function: The objective function of a traditional beam search algorithm is to find a hypothesis that has a maximum joint probability. If we directly apply the *vanilla* objective function on the constrained beam search algorithm, it is possible to generate sentences without containing any constraints by following the language model in GPT-2. To generate sentences that cover as many constraints as possible and are fluent, we propose a weighted objective function that returns a hypothesis with the

highest weighted score.

$$\hat{h} = \underset{h \in Hypotheses}{\operatorname{argmax}} \quad h.score + \beta \cdot \log\left(\frac{\sum(h.coverArray)}{|h.coverArray|} + \epsilon\right) \quad (9)$$

where *Hypotheses* represents a set of candidate hypotheses, $h.score$ denotes the log probability of hypothesis h , $h.coverArray$ is the *coverageArray* to track constraints in hypothesis h . β is a hyper-parameter to control the degree of influence of coverage ratio to hypothesis score. ϵ is a small number to avoid the coverage ratio to be zero ($\epsilon=1e-12$ in our experiments).

6 EXPERIMENTS

6.1 Data Collection

In our experiments, we use S&P 500 (SP) index data from 01/02/2001 to 03/13/2018. We split the whole data temporally where 2,898 days from 01/02/2001 to 07/10/2012 are reserved for training, 723 days from 07/11/2012 to 05/29/2015 are utilized for validation, and 702 days from 06/01/2015 to 03/13/2018 are used for the test set. Our task is to use the news titles to predict market volatility. WSJ is a U.S. finance-focused newspaper that has millions of news concerning the business. So, we crawl all the news in the WSJ archive⁵ from 2001 to 2018 and finally obtain 699,290 titles. We preprocess the dataset by taking advantage of the *article_type* tag and remove the titles that do not have relationships with the business. After preprocessing, 606,523 news titles are removed which leaves us with the final dataset of 92,767 titles.

6.2 Evaluation Metrics for Volatility Prediction

Firstly, to evaluate the performance of our models and baselines on volatility prediction, we employ Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE) and Mean Square Error (MSE) as evaluation metrics (lower MAPE, MAE and MSE are better).

In the financial field, investors care more about spikes, so we discretize and classify target variables into two classes and convert the target variable $Y \geq \mu + \sigma$ to the positive class and $Y < \mu + \sigma$ to the negative class where μ is the mean and σ is the standard error of target variable. Then, we use Precision, Recall, and F1-value as metrics to evaluate the models on predicting spikes.

6.3 Evaluation Metrics for Generation

To evaluate the quality of generated sentences, we define two metrics, Score and Utility. Score is used to assess the fluency of generated sentences and Utility is used to evaluate the coverage percentage of constraints (higher Score and Utility are better).

Given a hypothesis $h_n = \{y_0 \dots y_T\}$, $n \in [1, N]$, the *Score* and the *Utility* are defined as

$$Score = \frac{1}{N} \sum_n \left(\sum_t \log P(y_t | X; \{y_0 \dots y_{t-1}\}) \right) \quad (10)$$

$$Utility = \frac{1}{N} \sum_n \frac{\sum(h_n.coverArray)}{|h_n.coverArray|} \times 100\% \quad (11)$$

where T represents the length of sentence, y_t represents each word in sentence, and $t \in [1, T]$. N is the number of instances.

⁵<https://www.wsj.com/news/archive/>

Algorithm 3: Soft Constrained Dynamic Beam Allocation Algorithm

Input: Pretrained language model $GPT-2$; Constraint C
Parameters: Sentence Length L ; Beam size K ; Number of word to expand N
beam = Initialize_GPT-2(first_token);
ExtendConstraint = $\{\}$;
foreach subset in C **do**
 if subset has one token w **then**
 scores_list = cosineSimilarity(w, V); // Compute cosine similarity
 topN_indices = ARGMAX_N(scores_list); // The indices of top N tokens with high similarity
 ExtendConstraint[w] = topN_indices; // Expand single token in constraints
for $t = 1; t \leq L; t++$ **do**
 scores = GPT-2(beam); // scores is a $K \times |V|$ Matrix, where K is beam size, V is vocabulary
 constraints = [hyp.constraint for hyp in beam]; // Track the coverage of constraints in current hyp
 candidates = [(i, j , constraints[i].add(j)) for i, j in ARGMAX_K(K , scores)]; // Add the top K tokens into candidates
 for $h = 1; h \leq K; h++$ **do**
 foreach $w \in V$ that is unmet constraints for beam[h] **do**
 candidates.append((h, w , constraints[h].add(w))); // Add unmet constraints w into candidates
 if w is single word **then**
 for ext_w in ExtendConstraint[w] **do**
 candidates.append((h, ext_w , constraints[h].add(w))); // Add extended word and mark w as covered
 $w = \text{ARGMAX}(\text{scores}[h,:]);$
 candidates.append((h, w , constraint[h].add(w))); // For hypotheses h , add top 1 token over V into candidates
 selected $\leftarrow$ ALLOCATE(candidate, K); // Allocate beam
 beam $\leftarrow$ [candidates[i] for i in selected];
foreach hyp in beam **do**
 hyp_score = hyp.score() $\times \beta \cdot \log(\text{coverageRatio}(\text{hyp.constraints}) + \epsilon)$; // Weight the hypothesis score
BestHyp $\leftarrow$ hyp with highest hyp_score;
Output: BestHyp

Algorithm	Temperature	MaxLen	$C_{<\lambda_h}$		$C_{[\lambda_l, \lambda_h)}$		$C_{\geq \lambda_h}$	
			Score	Utility	Score	Utility	Score	Utility
DBA	0.8	30	-30.22	17.36%	-33.58	22.32%	-47.08	36.98%
DBA-TO	0.8	30	-31.68	9.76%	-51.06	31.92%	-62.80	47.44%
SC-DBA($\beta = 40$)	0.8	30	-44.66	72.77%	-52.15	64.77%	-66.54	78.34%
SC-DBA($\beta = 50$)	0.8	30	-48.34	77.83%	-55.43	69.21%	-69.45	81.81%
SC-DBA($\beta = 60$)	0.8	30	-50.08	80.59%	-57.26	71.42%	-70.05	82.72%
SC-DBA($\beta = 100$)	0.8	30	-50.08	80.59%	-60.36	74.31%	-70.45	83.23%
Sentences baseline	0.8	30	-179.41	12.08%	-170.31	16.86%	-164.19	9.56%
Constraints baseline	0.8	-	-48.83	100.00%	-65.72	100.00%	-72.11	100.00%

Table 2: Comparison between the baselines and SC-DBA under different parameters. Sentence threshold γ_{sent} is 0.5, word threshold γ_{word} is 0.2, the beam size is 10, β is a hyper-parameter to adjust the weighted objective function.

6.4 Baselines

For volatility, we compare our models with three baselines.

- NSVM [16] is a stochastic recurrent neural network framework which can formulate the temporal dynamics of volatility over time and predict the log-return $x_t = \log(s_t/s_{t-1})$ of actual prices s_t on date t . In contrast to our models, NSVM is evaluated based on Negative Log-Likelihood (NLL). To compare our models with NSVM on predicting the SPVt1, we train the model by multi-task learning. Since Σ_t^x represents the variance of x_t , SPVt1 represents the volatility of series, it can be regarded as variance as well, so we add another linear layer on the top of Σ_t^x and predict the SPVt1. The joint loss

is the sum of MSE and NLL. The dimensions of observable variables and the latent variables are same as the paper [16].

- StockNet [23] is a model to predict the movement of the stock based on tweets and historical prices. To compare the StockNet with our models, we have to modify the architecture: (1) StockNet predicts stock movements based on tweets and historical prices. In contrast, our models predict the volatility only based on news. So, we remove Market Information Encoder(MIE) component in StockNet and directly use our encoded input to replace MIE; (2) Since our models work on the regression problem, however, StockNet is applied on the classification task. We remove the softmax function on

Models	Fluency	Inform	Harmonic
DBA	4.744526	1.576642	2.36678378
DBA-TO	4.729927	1.532847	2.31534921
SC-DBA ($\beta = 50$)	4.416058	2.204380	2.94079332
Sentence baseline	3.007299	2.605839	1.32905602
Constraints baseline	1.160584	1.554745	2.79221249

Table 3: Human evaluation results.

the top layer so that the model can predict volatility; (3) We replace the likelihood term in objective function with MSE.

- Mean baseline: The mean of all true values in the training and validation dataset is the prediction.

We also compare our SC-DBA algorithm with four baselines based on score and utility of constraints,

- Dynamic Beam Allocation (DBA) [17]: DBA is a modification of beam search that yields decoder outputs user-supplied constraints. For a beam of size k and with C constraint token, DBA extends the hypotheses candidates to generate sentences with constraints tokens.
- DBA respects Temporal Order of constraints (DBA-TO): Since our algorithm respects the temporal order of constraint tokens, we modify the DBA to force the algorithm to generate the sentence considering the relative order of tokens.
- Constraint token baseline: We compute the log probability of constraint tokens, and the Utility of sentence is 100%.
- Sentence baseline: We filter the sentences whose sentence-level attention score is less than the threshold (pre-defined based on validation dataset) and concatenate the rest of sentences temporally. Then, we tokenize the sentence concatenation and cut it off by the maximum output length.

6.5 Results of Volatility prediction

Predicting market volatility is a challenging task. To demonstrate the relevance of our news-powered neural models, we first consider MAPE, MAE and MSE. As shown in Table 1 our models (except the MSE of Nested-Attention-LSTM and Nested-Attention-LSTM (Aug)) outperform all the baselines on SPVt1 prediction.

For predicting spikes, what stands out in Table 1 is that the precision, recall, and F1-value of Nested-Attention-LSTM (Aug) is higher than other models. Nested-Attention-LSTM (Aug) is the model using data augmentation technique. Hence, the experimental results, indeed, prove the capability of our augmentation technique.

6.6 Evaluation for Generation Algorithm

To evaluate the performance of the generation algorithms, we compare our generation algorithm with four baselines, quantitatively, in Table 2 and, qualitatively, (human evaluation) in Table 3. In Table 2 (Temperature controls the deterministic degree of the GPT-2, MaxLen is the maximum length of output), the first thing to notice is that utility of SC-DBA is higher than other baselines (except constraints baseline) under different parameters setting, which means the outputs of our algorithm can contain more constraint tokens. SC-DBA achieves a high coverage percentage of constraints, however, the log probability of generated sentences is increasing along

with the increment of β . High log probability does not mean that the algorithm will generate garbage sentences, it just does not follow the language model to generate sentences (If we apply *vanilla* beam search algorithm on GPT-2, the output will have a high probability, but the output is deterministic as well, which obeys our goal). Actually, GPT-2 can still generate fluent and meaningful sentences even if the log probability is high.

As evaluation of the performance of our generation algorithm and baselines is only based on the log probability of sentences, we also employ human evaluation to measure the generated sentences. We compile the instances from three classes into one file, and one instance has five sentences generated by five algorithms in Table 2.

To see if the outputs of the five algorithms contain financial information and are fluent, we conducted a human evaluation experiment where an annotator was shown the sentences generated by five algorithms (for each instance, the five sentences are shuffled, and their information such as they are generated by which algorithm was hidden to avoid bias from prior knowledge the annotator may have had). The annotator, who has experience with the financial field, was asked two questions: to what extent (on a scale of 1-5) they think the sentence is natural (Fluency); to what extent (on a scale of 1-5) they think the sentence has financial information (Informativeness). Overall, the annotator reviewed 685 sentences and answered 1,370 questions.

As shown in Table 3, the informativeness (Inform) of SC-DBA ($\beta=50$) is higher than other algorithms (except sentence baseline), which shows that the generated explanation of SC-DBA contains more financial information than other baselines. One of the possible reasons why sentence baseline has higher informativeness is that the sentences with high sentence-level self-attention scores have more financial information. Another finding is that the fluency of SC-DBA ($\beta = 50$) is not as good as DBA and DBA-TO which is consistent with the log probability and utility in Table 2. These results may be explained by the fact that following the language model will generate a natural language; meanwhile, it lost important constraint words led to the prediction.

To see the comprehensive performance of five algorithms, we also computed the harmonic mean of *Fluency* and *Informativeness* to measure the performance of algorithms, which balances fluency and informativeness. Obviously, SC-DBA has a higher harmonic mean, which means our algorithm can generate a more fluent and informative explanation. Also, we computed the *spearman correlation coefficient* of fluency and informativeness, which is -0.199; it shows that the fluency and informative has negative relationship under P-value 0.747. Although the relationship between fluency and informativeness is not significant, it is one more time to validate our observation that our algorithm can generate more informative sentences (contain more financial information). In contrast, even though the other two models (DBA and SC-DBA) have a higher fluency score, they do not carry too much useful information.

In Table 4, we also show the outputs of five different algorithms under the constraints (loss, oil, crude, price). As we can see, the DBA and DBA-TO generate a sentence that does not cover any constraint tokens by following the language model. In contrast, our algorithm can generate a sentence which shows the causality between loss of oil and crude price. Indeed, the traditional explanation only shows that the four words of loss, oil, crude prices

Algorithm	Sentences	Fluency	Informativeness
DBA	If you 're a fan of the original Star Wars trilogy , you 'll be happy to know that the new trilogy will be set in the same universe.	5	1
DBA-TO	If you 're a fan of the original Star Wars trilogy , you 'll be happy to know that the new trilogy will be set in the same universe.	5	1
SC-DBA ($\beta = 50$)	The loss of oil and gas production in the United States has been a major factor in the decline of the crude price .	5	5
Sentence baseline	DS&P downgrades Brazil for second time in less than six months Devon Energy to offer stock to boost liquidity Marathon Oil slashes capital spending amid first	3	3
Constraints baseline	loss oil crude price	1	2

Table 4: An example demonstrating the five generated sentences by five different algorithms.

trigger the model to predict volatility (y : 0.01447). In contrast, our model combines evidence and generates a synthetic event to tell us the reason for the prediction ($\hat{y}$: 0.01769, belongs class $C_{[\lambda_l, \lambda_h]}$). Moreover, being in class $C_{[\lambda_l, \lambda_h]}$ shows that loss, oil and crude price correlate to medium volatility. In summary, comparing with previous studies, our algorithm can connect each word along with the causal relationship and output a human-readable explanation.

7 CONCLUSION

In summary, we demonstrate the significance of our news-powered attention-based neural models to forecast the short-term volatility. Furthermore, we present a new data augmentation technique to overcome the data sparsity issue. Lastly, we propose the SC-DBA algorithm to generate human-readable explanations that outperforms baselines and can generate fluent explanations.

REFERENCES

- [1] Or Biran and Kathleen McKeown. 2017. Human-Centric Justification of Machine Learning Predictions. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 1461–1467.
- [2] Tim Bollerslev. 1986. Generalized Autoregressive Conditional Heteroskedasticity. *JOURNAL OF ECONOMETRICS* 31 (1986), 307–327.
- [3] Paula Branco, Rita P. Ribeiro, and Luis Torgo. 2016. UBL: an R package for Utility-based Learning. arXiv:1604.08079 [cs.MS]
- [4] Paula Branco, Luis Torgo, and Rita P. Ribeiro. 2017. SMOGN: a Pre-processing Approach for Imbalanced Regression. In *Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications (Proceedings of Machine Learning Research, Vol. 74)*, Luis Torgo, Bartosz Krawczyk, Paula Branco, and Nuno Moniz (Eds.). PMLR, ECML-PKDD, Skopje, Macedonia, 36–50.
- [5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- [6] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2014. Using Structured Events to Predict Stock Price Movement: An Empirical Investigation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1415–1425.
- [7] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep Learning for Event-Driven Stock Prediction. In *Proceedings of the 24th International Conference on Artificial Intelligence (Buenos Aires, Argentina) (IJCAI'15)*. AAAI Press, 2327–2333.
- [8] Robert F. Engle. 1982. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica* 50, 4 (1982), 987–1007. <http://www.jstor.org/stable/1912773>
- [9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780.
- [10] Chris Hokamp and Qun Liu. 2017. Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 1535–1546.
- [11] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The Curious Case of Neural Text Degeneration. CoRR abs/1904.09751 (2019). arXiv:1904.09751 <http://arxiv.org/abs/1904.09751>
- [12] J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019. Improved Lexically Constrained Decoding for Translation and Monolingual Rewriting. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 839–850.
- [13] J. Edward Hu, Rachel Rudinger, Matt Post, and Benjamin Van Durme. 2019. Para-Bank: Monolingual Bitext Generation and Sentential Paraphrasing via Lexically-constrained Neural Machine Translation. In *AAAI*.
- [14] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. 2017. Listening to Chaotic Whispers: A Deep Learning Framework for News-oriented Stock Trend Prediction. <https://doi.org/10.1145/3159652.3159690>
- [15] John C Hull and Alan White. 1987. The Pricing of Options on Assets with Stochastic Volatilities. *Journal of Finance* 42, 2 (1987), 281–300.
- [16] Rui Luo, Weinan Zhang, Xiaojun Xu, and Jun Wang. 2018. A Neural Stochastic Volatility Model. *AAAI Conference on Artificial Intelligence* (2018).
- [17] Matt Post and David Vilar. 2018. Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 1314–1324.
- [18] Alec Radford. 2018. Improving Language Understanding by Generative Pre-Training.
- [19] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).
- [20] Luis Torgo, Rita Ribeiro, Bernhard Pfahringer, and Paula Branco. 2013. SMOTE for Regression, Vol. 8154. 378–389. https://doi.org/10.1007/978-3-642-40669-0_33
- [21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5998–6008.
- [22] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. ArXiv abs/1910.03771 (2019).
- [23] Yumo Xu and Shay B. Cohen. 2018. Stock Movement Prediction from Tweets and Historical Prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 1970–1979. <https://doi.org/10.18653/v1/P18-1183>