

Realised Volatility Forecasting: Machine Learning via Financial Word Embedding

Eghbal Rahimikia, Stefan Zohren, and Ser-Huang Poon*

July 28, 2021

Abstract

We develop *FinText*, a novel, state-of-the-art, financial word embedding from Dow Jones Newswires Text News Feed Database. Incorporating this word embedding in a machine learning model produces a substantial increase in volatility forecasting performance on days with volatility jumps for 23 NASDAQ stocks from 27 July 2007 to 18 November 2016. A simple ensemble model, combining our word embedding and another machine learning model that uses limit order book data, provides the best forecasting performance for both normal and jump volatility days. Finally, we use Integrated Gradients and SHAP (SHapley Additive exPlanations) to make the results more ‘explainable’ and the model comparisons more transparent.

Keywords: Realised Volatility Forecasting, Machine Learning, Natural Language Processing, Word Embedding, Explainable AI, Dow Jones Newswires, Big Data.

JEL: C22, C45, C51, C53, C55, C58

*Eghbal Rahimikia (corresponding author) (eghbal.rahimikia@manchester.ac.uk) is at the University of Manchester, Alliance Manchester Business School and the Oxford-Man Institute of Quantitative Finance, University of Oxford, Stefan Zohren (stefan.zohren@eng.ox.ac.uk) is at the Oxford-Man Institute of Quantitative Finance, University of Oxford and Ser-Huang Poon (ser-huang.poon@manchester.ac.uk) is at the University of Manchester, Alliance Manchester Business School. Our special thanks go to Yoichi Otsubo at the Alliance Manchester Business School who provided dataset for this study. All models are run on the computational shared facility of the University of Manchester. We must express our sincere appreciation to the IT services of the University of Manchester for their constant and continued support and providing the computational infrastructures for this study.

1 Introduction

Many studies have attributed news as a major contributor to volatility (Engle and Ng, 1993; Engle and Martins, 2020; Conrad and Engle, 2021). In recent years, researchers have shown an increased interest in using natural language processing (NLP) and machine learning (ML) methods to extract relevant information from textual data such as news. So far, however, despite dramatic successes in other fields, these new techniques have attracted very little attention from the finance and economic scholarly communities (Gentzkow et al., 2019).

This paper explores the use of news in realised volatility (RV) forecasting using a state-of-the-art word embedding approach. Instead of using pre-trained Google and Facebook word embeddings, we develop *FinText*¹, a purpose-built financial word embedding for financial textual analysis, based on the Dow Jones Newswires Text News Feed, which covers different news services around the world with added emphasis on financial news. To the best of our knowledge, this is the first comprehensive study that develops a financial word embedding and uses it with a convolutional neural network (CNN) for RV forecasting. Unlike the Loughran-McDonald (LM) dictionary approach (Loughran and McDonald, 2011) that solely relies on predefined sets of words for extracting sentiment, our method extracts a substantial amount of information from big financial textual dataset without using any manually predefined resources and model assumptions. Moreover, to analyse the effect of a specific token on volatility forecasts, we use Explainable AI methods to make the forecasting performance evaluation more transparent and understandable.

Most RV forecasting studies use historical RV as the primary source to predict the next-day volatility using a linear model. Heterogeneous autoregressive (HAR) models, which are simple but yet effective linear models for RV forecasting, were first introduced by Corsi (2009). The further development of the HAR-family of models continued with HAR-J (HAR with jumps) and CHAR (continuous HAR) of Corsi and Reno (2009), SHAR (semivariance-HAR) of Patton and Sheppard (2015), and HARQ model of Bollerslev et al. (2016). The study of Rahimikia and Poon (2020a) provides a valuable comparison of the HAR-family of models and shows that the CHAR model is the best performing model considering 23 NASDAQ stocks. It also supplements the CHAR model with limit order book (LOB) data and sentiment variables extracted from financial news. The resulting CHARx model shows that news and LOB data provide statistically significant improvement in RV forecasts. By separating normal volatility days and volatility jump days, Rahimikia and Poon (2020a) shows that the negative sentiment has a clear impact on improving the RV forecasting performance of normal volatility days. Although the aforementioned work has successfully demonstrated that adding more information from news data improves the RV forecasting performance, it has certain limitations in terms of just using linear regression for forecasting.

During the last decade, there has been a growing number of publications focusing on the theory and application of ML in financial studies. Recent evidence suggests that this group of models can outperform traditional financial models in portfolio optimisation (Ban et al., 2018), LOB models for short-term price predictions (Zhang et al., 2018; Sirignano and Cont, 2019; Zhang and Zohren, 2021), momentum strategies (Lim et al., 2019; Poh et al., 2021; Wood et al., 2021), estimation of stochastic

¹*FinText* word embeddings are available for download from rahimikia.com/fintext.

discount factor (Chen et al., 2020), equity premium prediction using newspaper articles (Adämmer and Schüssler, 2020), measuring asset risk premiums (Gu et al., 2020), return prediction (Jiang et al., 2020), classifying venture capitalists (Bubna et al., 2020), designing trading strategies (Zhang et al., 2020), latent factor modelling (Gu et al., 2021), hedge fund return prediction (Wu et al., 2020), bond return prediction (Bianchi et al., 2021), and return forecasting using news photos (Obaid and Pukthuanthong, 2021), to name a few. In the context of RV forecasting, Rahimikia and Poon (2020b) comprehensively examined the performance of ML models using big datasets such as LOB and news stories from Dow Jones Newswires Text News Feed for 23 NASDAQ stocks. They show that LOB data has very strong forecasting power compared to the HAR-family of models, and adding news sentiment variables to the dataset only improves the forecasting power marginally. More importantly, they find that this is only valid for normal volatility days, not volatility jumps. However, this study remains narrow in focus dealing only with sentiment extracted from the LM dictionary. The two principal limitations of the LM dictionary are that it does not consider the language complexities, and it is developed based on financial statements, not news stories. Except few studies focusing on a novel statistical model for sentiment extraction to predict asset returns (Ke et al., 2019), topic modelling (Bybee et al., 2020), designing a sentiment-scoring model to capture the sentiment in economic news articles (Shapiro et al., 2020), and developing word embedding for analysing the role of corporate culture during COVID-19 pandemic (Li et al., 2020); so far, there has been little focus on more advanced NLP models for financial forecasting. Much of the current trend on NLP focuses on word embedding (Mikolov et al., 2013), a more sophisticated word representation that paved the way for the modern textual oriented ML models.

As a major review of textual analysis in accounting and finance, Loughran and McDonald (2016) warns that these more complex ML models potentially add more noise than signal. Nevertheless, we set out to investigate the usefulness of a more advanced NLP structure for RV forecasting. Part of the aim of this study is to develop, for the first time, a financial word embedding, named *FinText*, and compare it with publicly available general word embeddings. Another objective of this research is to determine whether a word embedding, together with a convolutional neural network (CNN) solely trained on news data, is powerful enough for RV forecasting or not. Finally, as another important objective, this study shines new light on these debates through the use of Explainable AI methods to interrogate these ML models.

We show that our financial word embedding is more sensitive to financial context compared with general word embeddings from Google and Facebook. Using 23 NASDAQ stocks from 27 July 2007 to 18 November 2016, we observe that using just previous day news headlines; all developed NLP-ML models strongly outperform HAR-family of models, extended models in Rahimikia and Poon (2020a), and other proposed ML models in Rahimikia and Poon (2020b) for forecasting RV on volatility jump days. Furthermore, *FinText* word embedding was developed based on the Word2Vec algorithm and Skip-gram model marginally outperforms all other word embeddings. This finding highlights the potential importance of news in more advanced NLP and ML frameworks for detecting volatility jumps. Also, it captured some rich textual information that the traditional dictionary-based sentiment models cannot capture.

Our findings also show that political news headlines have lesser predictive power compared with stock-related news. Comprehensive robustness checks focusing on the different number of days for news aggregation, different filter (kernel) size, and making embedding layers trainable support these findings. Another important finding that emerges from this study is that a simple ensemble model, which is a combination of the best performing model trained on LOB data in Rahimikia and Poon (2020b) and the best performing NLP-ML model in this study can substantially outperform all traditional and ML models for both normal volatility and volatility jump days. This shows the importance of combining financial information and textual news information in a single forecasting model. Using Explainable AI methods, we show that words classified as negative (e.g., ‘loss’) in the LM dictionary could cause both an increase or a decrease in RV forecast. The Explainable AI measures can also analyse the impact of any word (or combination of words) on RV forecasting, which was not feasible in the dictionary approach.

This paper is structured into seven sections. Section 2 deals with the theory of word embedding, Word2Vec, and FastText algorithms. Section 3 introduces our word embedding called *FinText*. News preprocessing steps are covered in Subsection 3.1 and evaluation and representation of this proposed word embedding in Subsection 3.2. Section 4 gives a brief review of RV forecasting followed by related models in Subsection 4.1. The proposed NLP-ML model for RV forecasting is introduced in Subsection 4.2, and Explainable AI methods in Subsection 4.3. Next, Section 5 presents the findings of the research, focusing on the stock-related news in Subsection 5.1, hot political news in Subsection 5.2, a comparison of models in Subsection 5.3, a proposed ensemble model in Subsection 5.4 and Explainable AI results in Subsection 5.5. Section 6 looks at robustness checks and finally, Section 7 concludes with a discussion and future research directions.

2 Word Embedding

A word embedding is one of the most important recent developments in NLP, where a word representation is in the form of a real-valued vector such that words that are closer in the vector space are expected to be similar in meaning. Before the development of word embeddings, each token in an one-hot encoding was defined by a binary vector of zeroes except the index for that token in the vocabulary list. The vectors of any two tokens are orthogonal to each other. Hence, a word embedding is a better representation of semantics. Specifically, a word embedding (V) is a $M \times N$ matrix, where M is the dimension size, and N is the number of unique tokens in the vocabulary list W . Each token is represented by a vector of M values. In the literature, M is usually about 300. Word2Vec (Mikolov et al., 2013) and FastText (Bojanowski et al., 2017) are two of the most efficient algorithms for training word embeddings. Subsection 2.1 and Subsection 2.2 briefly review these two simple and efficient algorithms.

2.1 Word2Vec

Mikolov et al. (2013) proposed supervised learning models with Skip-gram and continuous bag-of-words (CBOW) log likelihoods for the Word2Vec algorithm as shown below:

$$L_{skip-gram} = \frac{1}{T} \sum_{t=1}^T \left[\sum_{j=-k \setminus 0}^k \log p(w_{t+j}|w_t) \right], \quad (1a)$$

$$L_{CBOW} = \frac{1}{T} \sum_{t=1}^T \left[\sum_{j=-k \setminus 0}^k \log p(w_t|w_{t+j}) \right], \quad (1b)$$

where T is the total number of tokens in the sequence $X = \{t_1, t_2, \dots, t_T\}$, k is the window size around the chosen token w_t , and $p(w_{t+j}|w_t)$ and $p(w_t|w_{t+j})$ are the probability of correct predictions of Skip-gram and CBOW models, respectively. In the Skip-gram, the input (middle) token is used to predict the context (surrounding tokens), whereas the context (surrounding) tokens are used to predict the middle token in CBOW. It is generally agreed that the faster CBOW is suited for training larger datasets, while the Skip-gram is more efficient for training smaller datasets. Both models aim to maximise the aggregate predictive probability in Equation (1a) and Equation (1b) based on a simple neural network architecture.

For both Skip-gram and CBOW, the softmax operation for calculating the conditional probability is defined as follows:

$$p_{skip-gram}(w_c|w_t) = \frac{\exp(u_{w_c}^T u_{w_t})}{\sum_{l=1}^N \exp(u_l^T u_{w_t})}, \quad (2a)$$

$$p_{CBOW}(w_t|w_c) = \frac{\exp(u_{w_t}^T \bar{u}_{w_c})}{\sum_{l=1}^N \exp(u_l^T \bar{u}_{w_c})}, \quad (2b)$$

where w_c and w_t are context and target tokens, u_{w_c} and u_{w_t} are the trainable vector of context token w_c and target token w_t , and N is the number of tokens in the vocabulary. For the CBOW model in Equation (2b), as there are more than one context token, the average of the context token vectors, \bar{u}_{w_c} , is used. Both models are trained using stochastic gradient descent.

When there are a large number of tokens in the dictionary (N), Equation (2a) and Equation (2b) are computationally expensive. In this case, hierarchical softmax (Morin and Bengio, 2005) can be used instead. Another alternative is the negative sampling method with a binary logistic regression (Mikolov et al., 2013). In this method, the training set is the pair of target and context tokens, and K tokens are randomly chosen from a specific distribution. The output is one for the first token pair and zeroes for all other pairs. Mikolov et al. (2013) found that K ranges from 5 to 20 for large training sets and ranges from 2 to 5 for small training sets.

2.2 FastText

FastText is an extension of Word2Vec. For example, take ‘profit’ as a token and set n-grams equal to 3 (i.e. $n = 3$); the corresponding token vector is defined as $\langle \text{pr, pro, rof, ofi, fit, it} \rangle$, where \langle and \rangle

indicate the start and the end of the token vector. The original token $\langle \text{profit} \rangle$ is also added to this list. More formally, for token w , $u_w = \sum_{i=0}^n u_i$ where u_w is the vector of token w , n is the number of n-grams of this token, and u_i is the vector of each sub-token. In this enhanced representation, each token consists of a bag of n-gram characters. This algorithm is computationally intensive, but compared with Word2Vec algorithm, it is more powerful for learning rare and out-of-vocabulary (OOV) tokens and morphologically rich languages (MRL) (Bojanowski et al., 2017).

3 Financial Word Embedding: *FinText*

At the time of writing, several well-known pre-trained word embeddings are available. These include Mikolov et al. (2018) three-million-unique-tokens Word2Vec algorithm trained using the Google news dataset with about 100 billion words, and Joulin et al. (2016) one-million-unique-tokens FastText algorithm trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset. It is arguable whether these general word embeddings models are accurate for specialist financial forecasting. Some words will have a very different meaning when used in a specific financial context, e.g. apple as fruit and Apple as the technology company. To address this concern, we train a new word embedding, called *FinText*, using the Dow Jones Newswires Text News Feed database from 1 January 2000 to 14 September 2015. While specialising in corporate news, this big textual dataset is among the best textual databases covering finance, business, and political news services worldwide. Subsection 3.1 describes the database, the preprocessing steps and the properties of the word embedding developed from it, and Subsection 3.2 compares our *FinText* with those from Google and Facebook’s WikiNews mentioned above.

3.1 News Data, Preprocessing Steps and Model Properties

This study uses all types of news (viz. financial, political, weather, etc.) from Dow Jones Newswires Text News Feed from January 1, 2000, to September 14, 2015. All duplicate news stories and stories without headline and body are removed. Extensive text preprocessing of the news stories is required to eliminate redundant characters, sentences, and structures. Table A1 in the Appendix presents a brief review of the cleaning rules applied. Each rule is defined by a regular expression and may contain different variations. For brevity, only one variation is shown in this table. The text cleaning procedures fall into five main categories: 1) Primary, 2) Begins with, 3) Ends with, 4) General, and 5) Final checks. ‘Primary’ extracts the body of news from the extensible markup language (XML), removing XML-encoding characters (XMLENCOD), converting XML to text (parsing), converting uppercase to lowercase letters, and removing tables. ‘Begins with’ and ‘Ends with’ remove, respectively, parts begins and ends with the specified structures. ‘General’ caters for patterns that may appear in any part of the news stories. Finally, ‘Final checks’ removes links, emails, phone numbers, short news (lower than 25 characters), and the leading and the trailing space(s). These five sets of rules are applied to news headlines and bodies separately. Due to the importance of numbers in accounting and finance, numerical values are kept.

After cleaning the dataset, tokenisation breaks the headlines and news bodies into sentences and words. Common bigram (two-word) phrases are detected and replaced with their bigram form. All

tokens with less than five times of occurrences are ignored, the proposed bigram scoring function in Mikolov et al. (2013) is used with ten as the threshold value, and the maximum vocabulary size is set to 30 million to keep memory usage under control. Finally, the ‘_’ character is used for glueing pairs of tokens together. For example, if ‘financial’ and ‘statement’ appear consecutively exceeding a threshold number, they are replaced by ‘financial_statement’ as a new token. Altogether, *FinText* consists of 2,733,035 unique tokens. Following the preprocessing steps, Word2Vec and FastText algorithms are applied with window size, minimum count¹, negative sampling², and the number of iterations³ all set equal to 5. The initial learning rate (alpha) is 0.025, the minimum learning rate is 0.0001, and the exponent for negative sampling distribution is 0.75. Also, the dimension of word embeddings is 300. All these parameter values are the proposed ones by their developers.

3.2 Evaluation & Representation

The results for Skip-gram and CBOW models are reported for Word2Vec and FastText algorithms. These results are compared with pre-trained word embeddings from Google⁴ (Word2Vec algorithm) and Facebook⁵ (FastText algorithm).

3.2.1 General Purpose Predictive Accuracy

Table 1 compares all choices of word embedding based on the Google analogy benchmark. *FinText* is our financial word embedding developed using Dow Jones Newswires Text News Feed database, ‘Google’ is the Skip-gram word embedding trained on Google news dataset, and ‘WikiNews’ is Facebook’s Skip-gram word embedding trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset. Each section in the Google analogy benchmark contains a group of analogies. For example, under the ‘capital-common-countries’ section, the word embedding is challenged with the question, ‘London to England is like Paris to?’. The accuracy of each word embedding is reported for each section and all sections (overall).

From Table 1, it is apparent that, except for ‘currency’ and ‘gram1-adjective-to-adverb’, Facebook’s WikiNews has the highest predictive accuracy. The overall evaluation scores confirm this finding. The individual and overall scores for *FinText* suggest Skip-gram works better than CBOW. For general purpose word embedding, our financial word embedding is outperformed by Google under Word2Vec and outperformed by WikiNews under FastText. Table 6 presents the predictive accuracy based on the gold-standard collections, viz. WordSim-353 (Agirre et al., 2009) for measuring word relatedness and similarity, and Simlex (Hill et al., 2015) that focuses on similarity. All collections contain human-assigned judgements about the relatedness and similarity of word pairs. Performance is measured by Spearman’s rank correlation coefficient. It is apparent from this table that Google’s Word2Vec outperformed under WordSim-353, WikiNews outperformed under FastText; both outperformed *FinText* in their respective categories. As in the previous table, for *FinText*, Skip-gram is marginally better than CBOW.

¹The model ignores tokens with lower frequency than this value.

²Number of added noisy tokens matched with each chosen token.

³Number of epochs over the corpus.

⁴Word2Vec algorithms are downloadable from <https://code.google.com/archive/p/word2vec/>

⁵FastText algorithms are downloadable from <https://fasttext.cc/>

Table 1: Word embedding comparison (Google analogy)

Section	Word2Vec ^a			FastText		
	FinText ^b (CBOW) ^c	FinText (skip-gram)	Google (skip-gram)	WikiNews (skip-gram)	FinText (skip-gram)	FinText (CBOW)
capital-common-countries	77.27	85.50	83.60	100	85.93	47.40
capital-world	63.60	75.87	82.72	98.78	71.06	35.79
currency	22.49	36.69	39.84	25.00	32.54	10.65
city-in-state	19.93	60.48	74.64	81.41	58.20	15.83
family	63.46	70.51	90.06	98.69	58.97	59.62
gram1-adjective-to-adverb	27.47	33.00	32.27	70.46	50.59	79.45
gram2-opposite	33.33	32.50	50.53	73.91	50.83	71.67
gram3-comparative	77.65	75.04	91.89	97.15	77.06	87.39
gram4-superlative	61.67	55.00	88.03	98.68	62.14	90.71
gram5-present-participle	62.30	61.24	79.77	97.53	70.63	76.06
gram6-nationality-adjective	88.11	93.23	97.07	99.12	94.05	79.05
gram7-past-tense	42.02	39.92	66.53	87.25	37.98	31.09
gram8-plural	59.23	62.46	85.58	98.69	70.92	79.54
gram9-plural-verbs	53.26	54.53	68.95	97.38	61.59	79.17
overall	53.65	62.86	77.08	91.44	65.00	55.74

^a For learning word embeddings from textual datasets, **Word2Vec** is developed by Mikolov et al. (2013) and **FastText**, as an extension to Word2Vec algorithm, is developed by Bojanowski et al. (2017). ^b Developed word embedding on Dow Jones Newswires Text News Feed database (**FinText**); Publicly available word embedding trained on a part of Google news dataset with about 100 billion words (**Google**); Publicly available word embedding trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset (Mikolov et al., 2018) (**WikiNews**). ^c The continuous bag of words (CBOW) and Skip-gram are the proposed supervised learning models for learning distributed representations of tokens in Mikolov et al. (2013).

Table 2: Word embedding comparison (Gold-standard collections)

Benchmark	Word2Vec ^a			FastText		
	FinText ^b (CBOW) ^c	FinText (skip-gram)	Google (skip-gram)	WikiNews (skip-gram)	FinText (skip-gram)	FinText (CBOW)
WordSim-353 ^d (relatedness)	0.3821	0.4993	0.6096	0.6018	0.4425	0.1677
WordSim-353 (similarity)	0.6126	0.6436	0.7407	0.6713	0.6393	0.4722
Simlex	0.2657	0.2650	0.3638	0.3985	0.2772	0.2574

^a For learning word embeddings from textual datasets, **Word2Vec** is developed by Mikolov et al. (2013) and **FastText**, as an extension to Word2Vec algorithm, is developed by Bojanowski et al. (2017).

^b Developed word embedding on Dow Jones Newswires Text News Feed database (**FinText**); Publicly available word embedding trained on a part of Google news dataset with about 100 billion words (**Google**); Publicly available word embedding trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset (Mikolov et al., 2018) (**WikiNews**). ^c The continuous bag of words (CBOW) and Skip-gram are the proposed supervised learning models for learning distributed representations of tokens in Mikolov et al. (2013). ^d WordSim-353 (Agirre et al., 2009) is a gold-standard collection for measuring word relatedness and similarity, and Simlex (Hill et al., 2015) is another gold-standard collection tending to focus on similarity rather than relatedness or association.

Table 3: Financial analogy examples

Analogy	Word embedding		
	Google	WikiNews	FinText ^a
debit:credit :: positive:X	positive	negative	negative
bullish:bearish :: rise:X	rises	rises	fall
apple:iphone :: microsoft:X	windows_xp	iphone	windows
us:uk :: djia:X	NONE ^b	NONE	ftse_100
microsoft:msft :: amazon:X	aapl	hmv	amzn
bid:ask :: buy:X	tell	ask-	sell
creditor:lend :: debtor:X	lends	lends	borrow
rent:short_term :: lease:X	NONE	NONE	long_term
growth_stock:overvalued :: value_stock:X	NONE	NONE	undervalued
us:uk :: nyse:X	nasdaq	hsbc	lse
call_option:put_option :: buy:X	NONE	NONE	sell

^a *FinText* is the financial word embedding developed using Dow Jones Newswires Text News Feed database, Word2Vec algorithm and Skip-gram model. ^b Not in the vocabulary list.

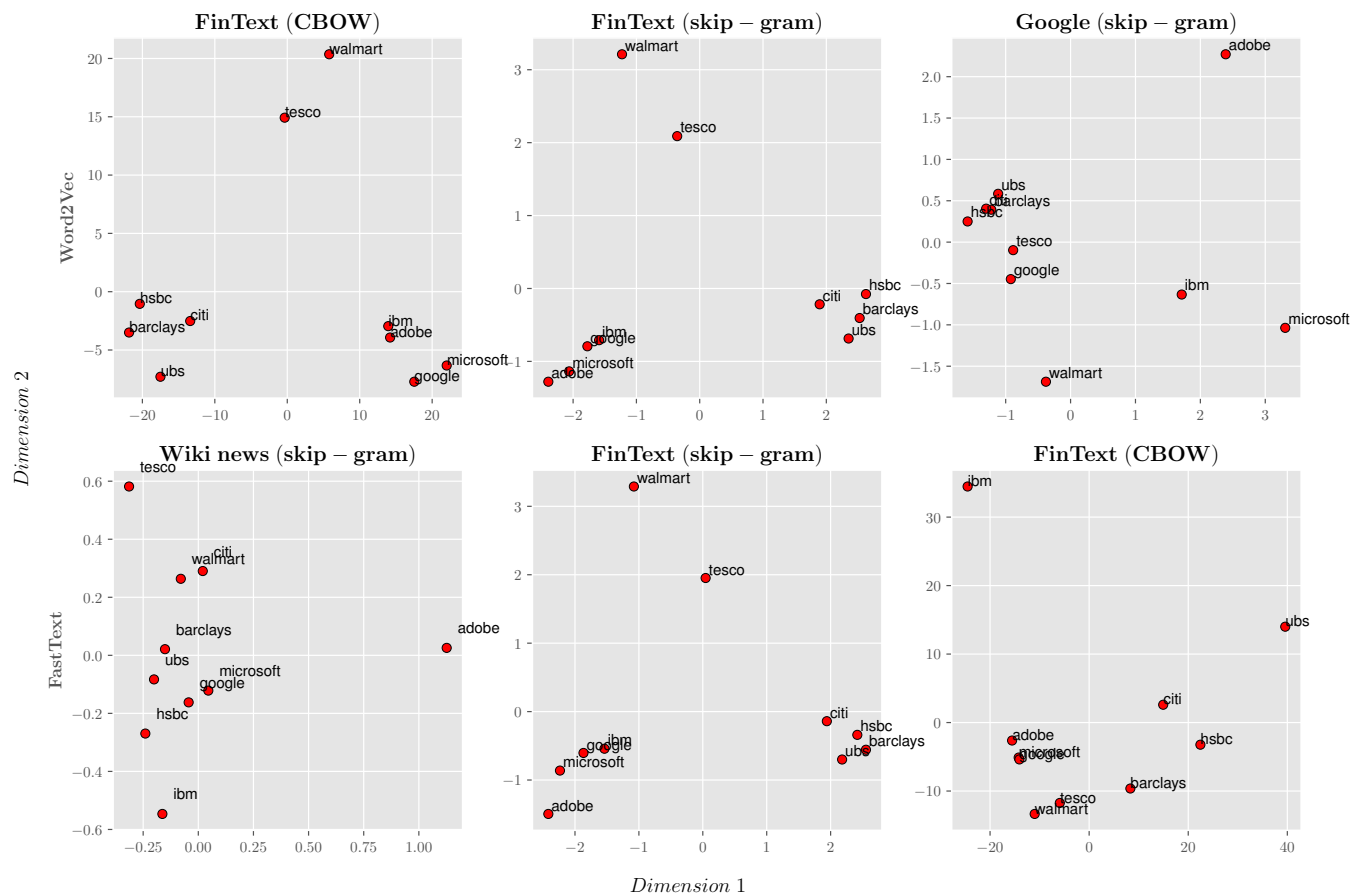


Figure 1: 2D visualisation of word embeddings

Notes: This figure shows the 2D visualisation of word embeddings. For each word embedding, principal component analysis (PCA) is applied to 300-dimensional vectors. The chosen tokens are ‘microsoft’, ‘ibm’, ‘google’, and ‘adobe’ (technology companies), ‘barclays’, ‘citi’, ‘ubs’, and ‘hsbc’ (financial services and investment banking companies), and ‘tesco’ and ‘walmart’ (retail companies). *Dimension 1* (x-axis) and *Dimension 2* (y-axis) show the first and second obtained dimensions. Word2Vec and FastText algorithms are shown in the first and second rows. FinText is the trained word embedding on Dow Jones Newswires Text News Feed database, Google is a publicly available developed word embedding trained on a part of Google news dataset, and WikiNews is another publicly available developed word embedding trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset. The continuous bag of words (CBOw) and Skip-gram are the proposed supervised learning models for learning distributed representations of tokens in Mikolov et al. (2013).

3.2.2 Financial Purpose Predictive Accuracy

For each word embedding, principal component analysis (PCA) is applied to the 300-dimensional vectors. Figure 1 presents the 2D visualisation of word embeddings. The tokens are chosen from groups of technology companies ('microsoft', 'ibm', 'google', and 'adobe'), financial services and investment banks ('barclays', 'citi', 'ubs', and 'hsbc'), and retail businesses ('tesco' and 'walmart'). *Dimension 1* (x-axis) and *Dimension 2* (y-axis) show the first and second obtained dimensions. Word2Vec is shown in the top row, and FastText is shown in the bottom row. Figure 1 shows that only *FinText* clusters all groups correctly, with Word2Vec producing generally better results than FastText.

Next, we challenged all word embeddings to produce the top three tokens that are most similar to 'morningstar'¹. This token is not among the training tokens of Google. WikiNews's answers are 'daystar', 'blazingstar', and 'evenin'. Answers from *FinText* (word2vec/skip-gram) are 'researcher_morningstar', 'tracker_morningstar', and 'lipper'². When asked to find unmatched token in a group of tokens such as ['usdgbp', 'euraud', 'usdcad'], a collection of exchange rates mnemonics, Google and WikiNews could not find these tokens, while *FinText* (word2vec/skip-gram) produces the sensible answer, 'euraud'.

Many word embeddings are able to solve word analogies such as king:man :: woman:queen.³ Table 3 lists some challenges we posed and the answers produced by the group of word embeddings considered here. While we do not have a financial gold standard collection, it is clear that our financial word embedding is more sensitive to financial contexts and able to capture very subtle financial relationships. It is beyond the scope of this paper to test these word embeddings rigorously because the focus here is on realised volatility forecasting. We will leave the investigation of the robustness of *FinText* to future research.

4 NLP-ML for Realised Volatility Forecasting

The previous section has illustrated that the *FinText* word embeddings developed in this paper are more sensitive to financial relationships. Here we aim to use these embeddings in the context of volatility forecasting. Already Engle and Ng (1993) and Engle and Martins (2020) show that news is a potential contributor to volatility. Therefore, we use word embeddings as a part of an ML model, called NLP-ML, to see if word embeddings are useful in forecasting realised volatility or not. In particular, Subsection 4.1 gives a brief review of RV forecasting, Subsection 4.2 presents NLP-ML, and Subsection 4.3 introduces Explainable AI.

4.1 Realised Volatility Forecasting

Assume an asset price P_t follows the stochastic process below:

$$d\log(P_t) = \mu_t dt + \sigma_t dw_t, \quad (3)$$

¹Morningstar is an American financial services firm was founded in 1984.

²Thomson Reuters Lipper is an American financial services firm was founded in 1973.

³':' means 'is to' and '::' means 'as'.

where μ_t is the drift, w_t is the standard Brownian motion, and σ_t is the volatility process (càdlàg function). RV, defined below, is used as a proxy for the unobserved integrated variance, $IV_t = \int_{t-1}^t \sigma_s^2 ds$:

$$RV_t \equiv \sum_{i=1}^M r_{t,i}^2, \quad (4)$$

where $M = \frac{1}{\delta}$ is the sampling frequency and $r_{t,i} \equiv \log(P_{t-1+i\delta}) - \log(P_{t-1+(i-1)\delta})$.

To date, the HAR-family of models is the most popular group of econometric models for forecasting RV. All HAR models follow the general specification below:

$$RV_{t+1} = f(\overline{RV}_{t-i}, J_t, \overline{BPV}_{t-i}, RV_t^{+/-}, \overline{RQ}_{t-i}), \quad (5)$$

where RV_{t+1} is the forecasted RV, \overline{RV}_{t-i} is the average RV of the last i days, J_t is the jump component¹, \overline{BPV}_{t-i} of day t is the average Bi-Power Variation (BPV) of the last i days, $RV_t^{+/-}$ is the positive/negative intraday return², \overline{RQ}_{t-i} is the average realised quarticity³ of the last i days, and f is a linear regression. Rahimikia and Poon (2020a) analysed 23 NASDAQ stocks and found the CHAR model to be the best performing HAR model. In Equation (5), the variable i for the BPV term in the CHAR model are the previous day ($i = 1$), the average of last week ($i = 7$), and the average of last month ($i = 21$) (Corsi and Reno, 2009).

In this study, consistent with Rahimikia and Poon (2020a) and Rahimikia and Poon (2020b), the training period is from 27 July 2007 to 11 September 2015 (2046 days), and the out-of-sample period is from 14 September 2015 to 18 November 2016 (300 days). RV is calculated during the NASDAQ market trading hours (from 9:30 AM to 4:00 PM Eastern Time). As a forecasting procedure, the rolling window method is applied in this study. Also, the LOB data from *LOBSTER* is used to calculate RV after applying the cleaning steps described in these studies. The RV descriptive statistics of 23 NASDAQ stocks are presented in Table 4. These tickers are chosen based on their liquidity (high to low) and availability of data. The sample period is from 27 July 2007 to 18 November 2016. Apart from a few exceptions (e.g., NVDA and MU), as liquidity decreases, the RV becomes more positively skewed with a slightly higher mean.

4.2 NLP-ML Structure

Figure 2 is an abstract representation of our NLP-ML model. $\{X_{(t,1)}, X_{(t,2)}, \dots, X_{(t,k_t)}\}$ is the vector of k_t tokens from news headlines on day t . When k_t is less than 500, the padding process will fill the vector to 500 with 'NONE' so that the daily input to the neural network has the same length. The reason for using only the news headline and not the news body is because putting all the news bodies together makes the sequence of tokens extremely long, even for just one day. A very long token

¹ $J_t = \text{Max}[RV_t - BPV_t, 0]$ is the jump at time t , where $BPV_t = \frac{1}{\mu_1^2} \sum_{i=1}^{M-1} |r_{t,i}| |r_{t,i+1}|$. M is the maximum value of sampling frequency, $r_{t,i}$ is the return at day t and sampling frequency i , and $\mu_1 = \sqrt{2/\pi}$ (Corsi and Reno, 2009).

² $RV_t^+ = \sum_{i=1}^M r_{t,i}^2 (r_{t,i} > 0)$ and $RV_t^- = \sum_{i=1}^M r_{t,i}^2 (r_{t,i} < 0)$ (Patton and Sheppard, 2015).

³ $RQ_t \equiv (\frac{M}{3}) \sum_{i=1}^M r_{t,i}^4$ (Bollerslev et al., 2016).

Table 4: RV descriptive statistics (from 27 July 2007 to 18 November 2016)

Ticker ^a	Min	Max	1 st quantile	Median	3 rd quantile	Mean	STD	Kurtosis	Skewness
AAPL	0.102	198.574	1.021	1.879	3.987	4.953	12.664	83.884	8.028
MSFT	0.067	133.679	0.935	1.577	2.918	3.289	7.268	104.476	8.786
INTC	0.146	130.167	1.165	1.920	3.709	4.065	7.596	71.609	6.875
CMCSA	0.148	153.376	0.951	1.827	3.751	4.014	8.419	95.178	8.139
QCOM	0.122	373.543	0.959	1.872	3.873	4.673	13.778	280.384	13.730
CSCO	0.163	343.946	1.030	1.791	3.438	4.348	12.995	262.347	13.440
EBAY	0.215	252.608	1.461	2.592	4.946	5.525	12.785	139.670	9.850
GILD	0.222	259.489	1.383	2.179	3.900	4.719	13.706	173.238	11.815
TXN	0.183	287.897	1.111	1.999	3.987	4.107	9.249	398.325	15.651
AMZN	0.206	547.030	1.683	2.882	5.720	7.562	23.925	185.115	11.593
SBUX	0.164	192.629	1.086	1.968	4.308	4.714	11.255	114.155	9.331
NVDA	0.317	1104.351	2.180	4.382	9.414	9.591	29.432	837.584	24.558
MU	0.563	359.620	4.204	7.137	13.584	14.355	26.204	61.344	6.711
AMAT	0.292	114.376	1.715	2.812	5.150	5.153	8.149	61.231	6.438
NTAP	0.257	290.647	1.594	2.903	5.743	6.283	14.419	149.830	10.163
ADBE	0.216	569.720	1.153	2.081	3.952	5.059	15.730	693.479	21.309
XLNX	0.229	251.383	1.224	2.184	4.258	4.359	9.382	265.118	12.977
AMGN	0.159	214.156	1.006	1.727	3.126	3.468	9.764	221.110	13.295
VOD	0.134	219.033	0.780	1.487	3.369	4.252	11.788	115.204	9.471
CTSH	0.246	485.894	1.214	2.162	5.266	6.103	17.479	315.162	14.555
KLAC	0.154	499.808	1.278	2.514	5.126	5.689	17.915	395.464	17.684
PCAR	0.214	389.930	1.285	2.563	5.800	6.014	13.514	294.091	12.810
ADSK	0.358	693.772	1.637	2.823	5.256	6.833	24.263	413.773	17.814

^a Tickers are ranked according to their liquidity (high to low).

sequence increases the number of model parameters, stresses the computation, and could result in over-fitting, especially when our training sample size is relatively small. Also, it is often felt that the news headline is the most important abstract of the news body.

As shown in Figure 2, the word embedding section is separated into days with news and days without news. For days with news, each token $X_{(t,k_t)}$ has a 1×300 word embedding vector from one of the six pre-trained word embeddings in Section 3. These vectors are fixed and made non-trainable to reduce the number of parameters to be trained. So this results in a 500×300 sentence matrix to be fed into a CNN layer. On days when there is no news, the vector is initially filled with random numbers that can be trained by the neural network. After the CNN, we have a fully connected neural network (FCNN) that turns the signals into a single RV forecast, RV_{t+1} . Following Bollerslev et al. (2016), an ‘insanity’ filter is applied: For each rolling window, the minimum, maximum, and average of training RVs are calculated. Any RV forecast that is greater (smaller) than the maximum (minimum) value will be replaced by the rolling window average RV.

Figure 3 illustrates the structure of the CNN used in this study. Starting from the sentence matrix from Figure 2 for the news headline, ‘apple looks to be further beefing up siri.’, three filters of size 3, 4, and 5 are applied simultaneously with valid padding¹ and a stride size² of 1.³ The outputs are

¹In VALID padding (in contrast to SAME padding), the output feature map has a smaller vector size than the input word embedding vector size.

²Stride size defines the amount of filter movement over the word embedding vector.

³In the tuning process, we can choose how many sets of size 3, 4, and 5 filters to use. We have tested 3, 8, 13, 18, 23, 28, 33, 38, and 43 sets of filters in the empirical section.

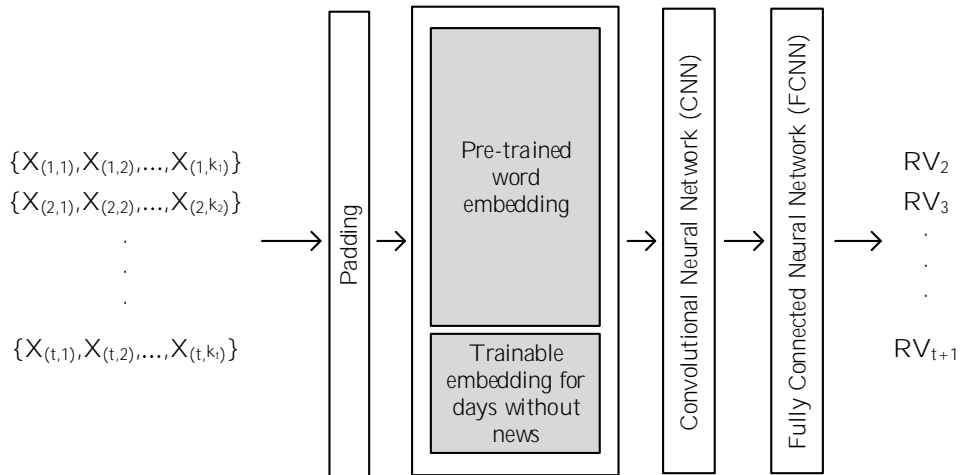


Figure 2: An abstract representation of NLP-ML model

Notes: $\{X_{(t,1)}, X_{(t,2)}, \dots, X_{(t,k_t)}\}$ consists of news headlines of day t and $X_{(t,k_t)}$ is the k^{th} token of input t . Also, RV_{t+1} is the RV of day $t+1$ (next day RV). Padding with the maximum length of 500 is adopted to ensure that all inputs of the neural network have the same length. The word embedding block consists of two different word embeddings. To capture days without any news, a trainable word embedding is used.

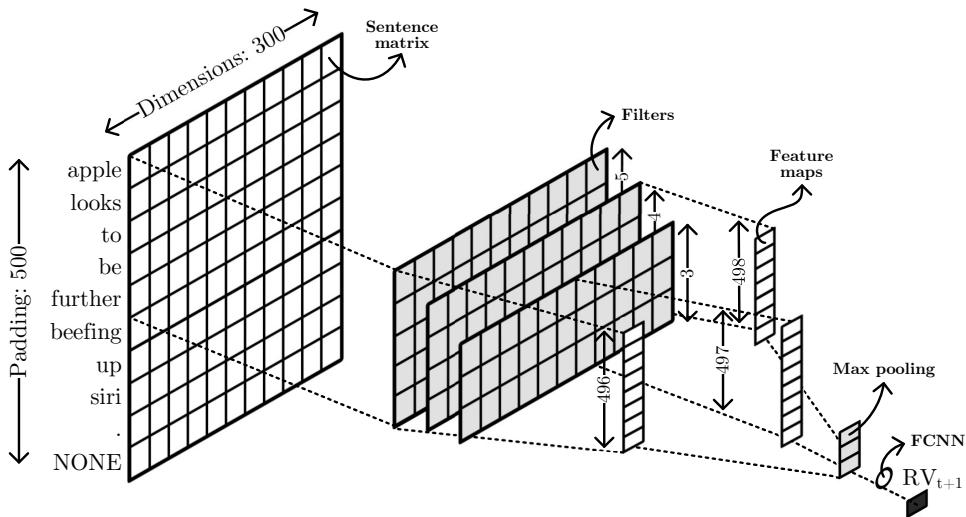


Figure 3: A detailed representation of NLP-ML model

Notes: The sentence matrix is a 500×300 matrix with a maximum length of padding of 500 and word embedding dimensions of 300. In this matrix, each token is defined by a vector of 300 values. This structure contains three filters of different sizes. The filters with the size of 3, 4, and 5 generate feature maps with the size of 498, 497, and 496, respectively. Max pooling and a fully connected neural network (FCNN) are applied then as the next steps. The output of this network is the RV of the next day (RV_{t+1}).

three 1-dimensional feature maps of sizes 498, 497, and 496. Specifically, following Kim (2014), let $X_i \in \mathbb{R}^M$ be the M -dimensional token vector corresponding to the i^{th} token in the news headline. We know from Figure 2 that $M = 300$, and news headlines with less than 500 tokens will be padded so that $n = 500$. Let $X_{i:i+j}$ refer to the concatenation of token vectors $X_i, X_{i+1}, \dots, X_{i+j}$ as follows:

$$X_{i:i+j} = X_i \oplus X_{i+1} \oplus \dots \oplus X_{i+j}, \quad (6)$$

where \oplus is the concatenation operator. A convolution operation involves a filter $W \in \mathbb{R}^{hM}$, which is applied to a window size of h tokens to produce a new feature as follow:

$$C_i = f(W \cdot X_{i:i+h-1} + b), \quad (7)$$

where $b \in \mathbb{R}$ is a bias term, and f is a nonlinear function. This filter is applied to each possible window of tokens in the sentence $x_{1:h}, x_{2:h+1}, \dots, x_{n-h+1:n}$ to produce a feature map with $C \in \mathbb{R}^{n-h+1}$,

$$C = \{C_1, C_2, \dots, C_{n-h+1}\}. \quad (8)$$

As the next step, max-pooling ($\hat{C} = \max\{C\}$) is applied. This step is used to ensure that the most important feature is chosen (Collobert et al., 2011). For converting the max-pooling layer to the RV of the next day (RV_{t+1}), an FCNN is used as the last layer. The activation function of both the CNN and the FCNN is a rectified linear unit (ReLU)¹, the optimisation algorithm is Adam (Kingma and Ba, 2014), and MSE is the objective function of this network. To prevent the NLP-ML model from over-fitting, L^2 regularisation with weight decay value set equal to 3 is used for both CNN and FCNN, while the dropout rate is set equal to 0.5 between the CNN and FCNN. The timespan for headlines of the day t starts from 9:30 AM Eastern Time of day t and ends at 9:30 AM Eastern Time of the day $t + 1$. Daily training of this model is computationally intensive; therefore, the training process is repeated every five days, and the trained model is used for the next days. In order to have reproducible results, a random number generator (RNG) with the same seed is used for all trained models.

Dow Jones Newswires Text News Feed provides a tagging system for finding news stories related to a specific stock. Therefore, in this study, considering the timespan of our analysis and availability of this tagging system during the timespan, the tag ('about') is used for extracting the stock-related news for each ticker. This tag denotes a story about a ticker but of no particularly significant. Also, for hot political news, news stories with a 'hot' tag and also 'politics' as the subject tag are used. 'Hot' tag means news story is deemed 'important' or 'timely' in some way. The main reason for adding 'hot' as an extra tag here is to reduce the length of daily tokens as much as possible. Returning to padding, Figure 4 shows the distribution of daily tokens (headlines) for stock-related news in Figure 4a and hot political news in Figure 4b. What can be clearly seen from these figures is that for hot political news, padding is removing more tokens compared to stock-related news. This is expected because the number of daily hot political news is usually higher than stock-related news.

¹Using Relu activation function for FCNN prevents the NLP-ML model from generating negative RVs.

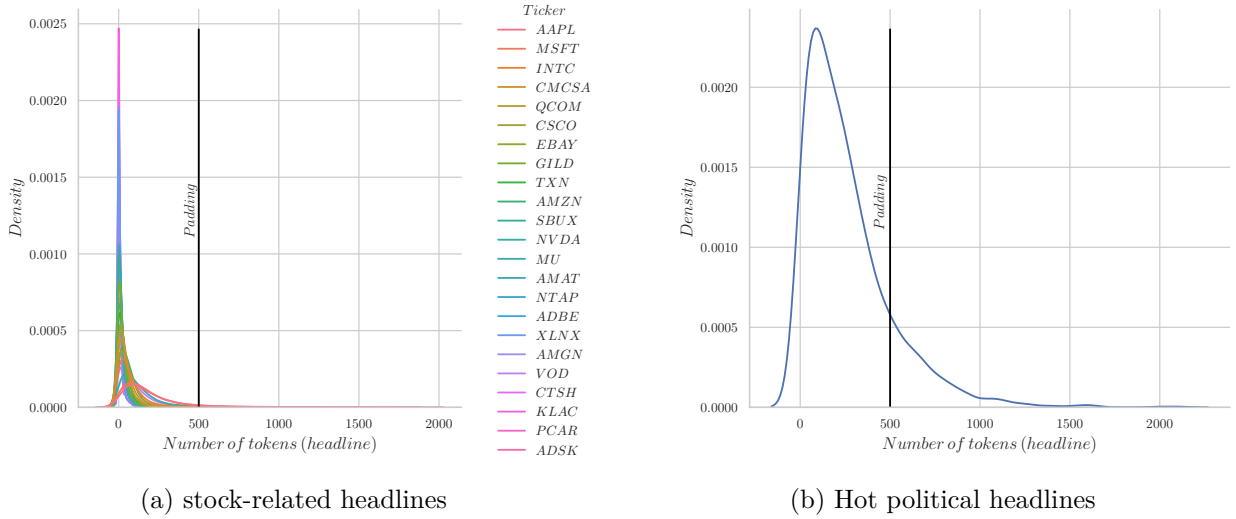


Figure 4: Distribution of daily tokens

Notes: Following putting daily stock-related news (left plot) and hot political (right plot) headlines together (train data - 2046 days), the number of daily tokens is calculated, and their distributions are plotted. The vertical line is the chosen maximum length of padding (500).

4.3 Explainable AI

This subsection describes two popular methods, the integrated gradient (IG) and the shapely additive explanations (SHAP), for making AI models more transparent. Sundararajan et al. (2017) defined IG with Riemann sum as follows:

$$IG_i(X) ::= \frac{(X_i - X'_i)}{m} \sum_{k=1}^m \frac{\partial F(X' + \frac{k}{m}(X - X'))}{\partial X_i}, \quad (9)$$

where IG_i is the integrated gradient for token i , X is the set of weights for the daily news headlines, X' is the baseline (i.e. sequence of zeros here), m is number of steps in the Riemann sum, $(X' + \frac{k}{m}(X - X'))$ is the linear interpolation between the baseline (X') and X , $F(\cdot)$ is the trained NLP-ML model, $\frac{\partial F(\cdot)}{\partial X_i}$ computes the gradient of the NLP-ML model with respect to each token i , $\frac{1}{m} \sum_{k=1}^m Gradients$ is the average gradient, and $(X_i - X'_i)$ is for scaling the integrated gradient. In this study, we use the Gauss-Legendre quadrature (with 50 steps, and a batch size of 100) instead of Riemann sum for the approximation.

The IG approach is applicable to any differentiable model. It has a theoretical foundation, it is easy to use, and is not computationally intensive. Moreover, IG provides the individual importance (attribute) for each token; a higher attribute value indicates a higher RV forecast and *vice versa* for a lower attribute value. Unfortunately, it is not possible to rank the tokens in order to find out which has the best predictive power for realised volatility.

Lundberg and Lee (2017) proposed the SHAP method based on the coalition game theory. Shapley values ϕ_i , defined below, show the importance of a model input S (a set of tokens in daily news

headlines) given the model output $f(S)$, the volatility forecast. In this case:

$$\phi_i = \frac{1}{|N|!} \sum_{S \subseteq N \setminus \{i\}} |S|! (|N| - |S| - 1)! [f(S \cup \{i\}) - f(S)], \quad (10)$$

where $f(S \cup \{i\}) - f(S)$ captures the marginal contribution in volatility forecast of adding token i to the set S , N contains all model inputs, $|S|!$ shows the number of different ways the chosen set of tokens may be presented, and $(|N| - |S| - 1)!$ is the number of different ways that the remaining tokens could have been added. The Shapley values ϕ_i show the magnitude and sign of the average contribution of token i ; it satisfies three properties, viz. local accuracy (additivity)¹, missingness (nonexistence or null effect)², and consistency (symmetry)³. As tokens are added to the set, changes in the RV forecast reflect their relevance.

The benefits of the SHAP approach include a solid theoretical foundation in game theory and no requirement for differentiable models. However, it is computationally intensive, and, like other permutation-based approaches, it does not consider feature dependencies and may generate misleading results. Here, we use a high-speed approximation algorithm, Deep SHAP based on DeepLIFT (Shrikumar et al., 2017), to calculate SHAP values.

5 Results

For each stock i , the performance difference between the NLP-ML model j and CHAR is measured as follow:

$$Avg \Delta_{MSE,i,j} = \frac{1}{23} \sum_{i=1}^{23} (MSE_i(NLP-ML_j) - MSE_i(CHAR)), \quad (11)$$

$$Med \Delta_{MSE,i,j} = Median(MSE_i(NLP-ML_j) - MSE_i(CHAR)). \quad (12)$$

In equations (11) and (12), MSE could be replaced by QLIKE⁴ (Patton, 2011) or mean directional accuracy (MDA). For MSE and QLIKE, a negative value in equations (11) and (12) indicates improvement, and a positive value indicates degradation in performance. Conversely, for MDA, a positive value indicates improvement, and a negative value indicates degradation in performance. Finally, a reality check (RC) is used for comparing each NLP-ML model against all models in the HAR-family (viz. AR1, HAR, HAR-J, CHAR, SHAR, ARQ, HARQ, and HARQ-F) as follows:

$$\begin{aligned} H_0 : Min_{k=1, \dots, n} \mathbb{E}[L^k(RV, X) - L^0(RV, X)] &\leq 0, \\ H_1 : Min_{k=1, \dots, n} \mathbb{E}[L^k(RV, X) - L^0(RV, X)] &> 0, \end{aligned} \quad (13)$$

¹It means that sum of the individual token attributions is equal to the forecasted RV.

²It means that a missing token has no attributed impact ($\phi_i = 0$).

³It means that if a change in a specific token has a more considerable impact on the first model compared to the second model, the importance of this token should be higher for the first model than the second model.

⁴QLIKE(RV_t, \widehat{RV}_t) $\equiv \frac{RV_t}{\widehat{RV}_t} - \log\left(\frac{RV_t}{\widehat{RV}_t}\right) - 1$ where RV_t and \widehat{RV}_t are the true and fitted RV at time t respectively. Patton (2011) showed that this loss function, MSE, and their variations are the only class of robust loss functions for ranking volatility forecasting models.

where L^k is the loss from the benchmark (HAR-family of models), L^0 is the loss from the specific NLP-ML model, and n is the number of benchmark models (in this study, $n = 8$). Rejection of H_0 means that the loss from the NLP-ML model is significantly smaller than that from all benchmark models. For this RC test, we follow the stationary bootstrap of Politis and Romano (1994) with 999 re-samplings and an average block length of 5 (Bollerslev et al., 2016; Rahimikia and Poon, 2020a,b).

Rahimikia and Poon (2020a) showed the importance of separating normal volatility days and volatility jump days when evaluating out-of-sample forecasting performance. A day is defined as a volatility jump day when RV for that day is greater than $Q3 + 1.5 IQR$, where $IQR = Q3 - Q1$, and $Q1$ and $Q3$ are, respectively, the first and third quantiles of RV. By applying this criterion to the sample of 23 stocks, averagely about 90% of the 300 days in the out-of-sample period are classified as normal volatility days, and the remaining 10% as volatility jump days.

5.1 Stock-Related News

Table 5 reports the the out-of-sample forecasting results for different NLP-ML models with 3, 8, 13, 18, 23, 28, 33, 38, and 43 CNN filters. An NLP-ML model with a lower number of filters is less complex than one with a higher number of filters. In Table 5, RC is the percentage of tickers with outstanding NLP-ML performance against all HAR-family of models at the 5% and 10% significant levels with MSE(QLIKE) as the loss function. ‘Avg’ and ‘Med’ are, respectively, $Avg \Delta_{MSE,i,j}$ and $Med \Delta_{MSE,i,j}$ in Section 5.

The top panel of Table 5 is for normal volatility days, while the bottom panel is for volatility jump days. The bottom panel clearly shows that NLP-ML dominates the HAR models on volatility jump days. Using the MSE loss function, our ‘FinText(skip-gram)’ and ‘FinText (FT¹/skip-gram)’ are the best performing word embeddings. The MDA results in Table A2 in the Appendix corroborates this finding. With QLIKE loss function, Google and WikiNews show slightly better results. On normal volatility days in the top panel, all NLP-ML models underperform and are dominated by HAR models.

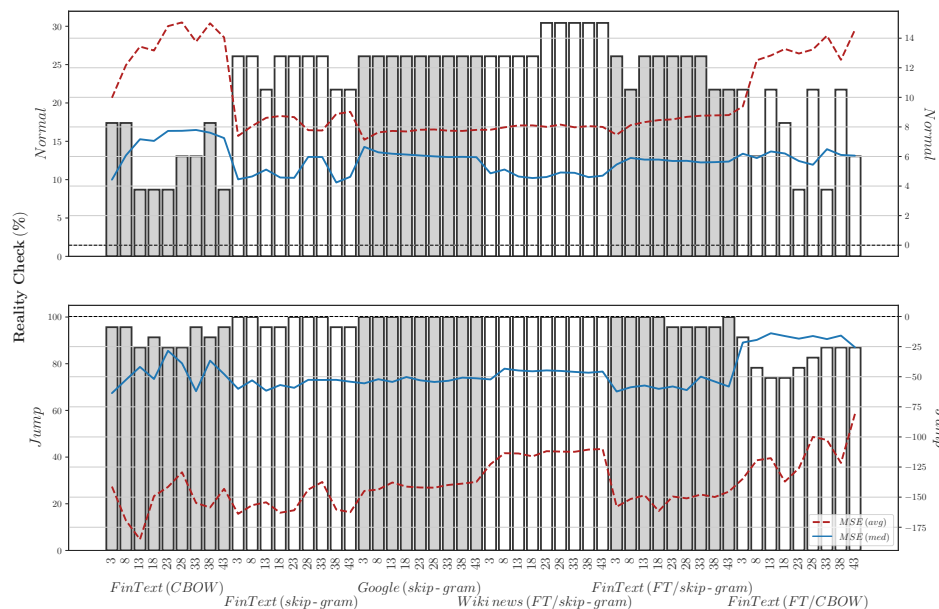
Figure 5 plots the results in Table 5. The top panel of Figure 5 is for normal volatility days while the bottom panel is for volatility jump days. The left column is based on MSE, while the right column is based on QLIKE. The solid line represents the median, while the dashed line represents the mean. A negative value of the loss function (either MSE or QLIKE) means NLP-ML outperforms, and a positive value means the CHAR model is better. The bar chart represents the proportion of RC tests that shows NLP-ML outperforms the HAR-family of models. A value over 50% means NLP-ML is better, and a value closer to 100% means NLP-ML dominates. Like Table 5, the most striking result in Figure 5 is the clear dominance of NLP-ML models in the bottom panel on volatility jump days. The Skip-gram result is more stable than CBOW, and, with the QLIKE loss function, CBOW performs badly compared with other word embedding choices. With the MSE loss function, our ‘FinText(skip-gram)’ and ‘FinText(FT/skip-gram)’ perform well and are marginally better than the other word embeddings. Interestingly, the number of CNN filters does not appear to have a high impact on the performance. This means one could choose a simpler model with a smaller number of CNN filters. The MDA loss function results in Figure A1 shows NLP-ML models are generally good

¹FastText algorithm.

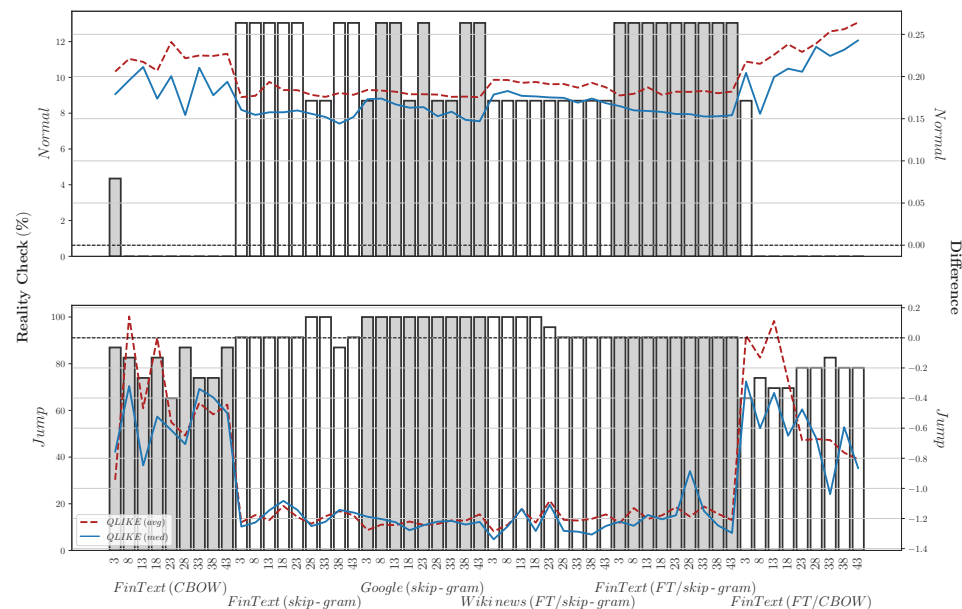
Table 5: Out-of-sample forecasting performance and RC results for NLP-ML models - stock-related news

Normal days	Filters	FinText (CBOW)									FinText (skip-gram)									Google (skip-gram)								
		3	8	13	18	23	28	33	38	43	3	8	13	18	23	28	33	38	43	3	8	13	18	23	28	33	38	43
MSE	Avg	9.97 ^a	12.18	13.43	13.17	14.82	15.07	13.76	15.01	14.05	7.39	8.06	8.61	8.74	8.66	7.78	7.76	8.87	9.02	7.13	7.63	7.73	7.69	7.81	7.83	7.74	7.73	7.80
	Med	4.43 ^b	6.08	7.17	7.05	7.73	7.74	7.79	7.61	7.24	4.45	4.65	5.12	4.58	4.56	5.97	5.96	4.24	4.63	6.65	6.28	6.19	6.13	6.07	6.01	5.96	5.98	5.95
RC	0.05	17.39 ^c	17.39	8.70	8.70	8.70	13.04	13.04	17.39	8.70	26.09	26.09	21.74	26.09	26.09	26.09	26.09	21.74	21.74	26.09	26.09	26.09	26.09	26.09	26.09	26.09	26.09	26.09
	0.10	30.44 ^c	30.44	34.78	26.09	30.44	30.44	26.09	26.09	21.74	26.09	26.09	26.09	26.09	26.09	26.09	26.09	26.09	26.09	30.44	30.44	30.44	26.09	30.44	26.09	26.09	30.44	30.44
QLIKE	Avg	0.21	0.22	0.22	0.21	0.24	0.22	0.22	0.22	0.23	0.18	0.18	0.19	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18
	Med	0.18	0.20	0.21	0.17	0.20	0.15	0.21	0.18	0.19	0.16	0.15	0.16	0.16	0.16	0.16	0.15	0.14	0.15	0.17	0.17	0.17	0.16	0.16	0.15	0.16	0.15	0.15
RC	0.05	4.35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	13.04	13.04	13.04	13.04	13.04	8.70	8.70	13.04	13.04	8.70	13.04	13.04	8.70	13.04	8.70	8.70	13.04	13.04
	0.10	4.35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	13.04	13.04	13.04	13.04	13.04	8.70	8.70	13.04	13.04	8.70	13.04	13.04	8.70	13.04	8.70	8.70	13.04	13.04
		WikiNews (FT ^d /skip-gram)									FinText (FT/skip-gram)									FinText (FT/CBOW)								
MSE	Avg	7.81	7.97	8.10	8.10	8.01	8.15	7.97	8.06	8.00	7.47	8.12	8.32	8.46	8.51	8.68	8.76	8.78	8.80	9.39	12.51	12.83	13.27	12.96	13.24	14.17	12.54	14.57
	Med	4.86	5.12	4.65	4.54	4.61	4.92	4.89	4.60	4.70	5.46	5.90	5.79	5.79	5.69	5.71	5.60	5.62	5.66	6.19	5.89	6.34	6.20	5.69	5.44	6.50	6.10	6.06
RC	0.05	26.09	26.09	26.09	26.09	30.44	30.44	30.44	30.44	30.44	26.09	21.74	26.09	26.09	26.09	26.09	26.09	21.74	21.74	21.74	13.04	21.74	17.39	8.70	21.74	8.70	21.74	13.04
	0.10	30.44	26.09	30.44	30.44	30.44	30.44	30.44	30.44	30.44	30.44	26.09	26.09	26.09	26.09	26.09	26.09	26.09	26.09	26.09	26.09	30.44	26.09	26.09	26.09	21.74	26.09	17.39
QLIKE	Avg	0.20	0.20	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.18	0.18	0.19	0.18	0.18	0.18	0.18	0.18	0.18	0.22	0.22	0.23	0.24	0.23	0.24	0.25	0.26	0.26
	Med	0.18	0.18	0.18	0.18	0.18	0.17	0.17	0.17	0.17	0.16	0.16	0.16	0.16	0.16	0.16	0.15	0.15	0.15	0.20	0.16	0.20	0.21	0.21	0.24	0.22	0.23	0.24
RC	0.05	8.70	8.70	8.70	8.70	8.70	8.70	8.70	8.70	8.70	13.04	13.04	13.04	13.04	13.04	13.04	13.04	13.04	13.04	8.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.10	8.70	8.70	8.70	8.70	8.70	8.70	8.70	8.70	8.70	13.04	13.04	13.04	13.04	13.04	13.04	13.04	13.04	13.04	8.70	0.00	4.35	0.00	0.00	0.00	0.00	0.00	0.00
Jumps		FinText (CBOW)									FinText (skip-gram)									Google (skip-gram)								
MSE	Avg	-141.22	-169.42	-185.01	-149.12	-141.51	-129.34	-154.83	-158.58	-143.06	-163.88	-156.68	-154.28	-162.94	-160.90	-143.73	-137.40	-160.28	-162.53	-144.85	-143.68	-137.84	-141.25	-141.99	-142.11	-139.87	-138.76	-137.31
	Med	-63.62	-52.57	-41.87	-51.90	-28.45	-38.90	-62.04	-36.67	-47.84	-60.06	-52.98	-61.62	-56.91	-59.21	-52.60	-52.69	-52.59	-54.14	-55.50	-52.06	-54.33	-50.25	-53.01	-54.37	-53.38	-50.77	-51.31
RC	0.05	95.65	95.65	86.96	91.30	86.96	86.96	95.65	91.30	95.65	100	100	95.65	95.65	100	100	100	95.65	95.65	100	100	100	100	100	100	100	100	100
	0.10	95.65	95.65	95.65	100	86.96	91.30	95.65	91.30	95.65	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
QLIKE	Avg	-0.94	0.14	-0.47	0.00	-0.56	-0.65	-0.43	-0.51	-0.44	-1.23	-1.18	-1.21	-1.12	-1.19	-1.23	-1.19	-1.15	-1.18	-1.28	-1.24	-1.24	-1.22	-1.24	-1.24	-1.21	-1.21	-1.17
	Med	-0.76	-0.32	-0.85	-0.52	-0.61	-0.71	-0.34	-0.40	-0.50	-1.25	-1.23	-1.15	-1.08	-1.14	-1.25	-1.22	-1.14	-1.16	-1.19	-1.20	-1.22	-1.28	-1.24	-1.22	-1.22	-1.24	-1.22
RC	0.05	86.96	82.61	73.91	82.61	65.22	86.96	73.91	73.91	86.96	91.30	91.30	91.30	91.30	91.30	100	100	86.96	91.30	100	100	100	100	100	100	100	100	100
	0.10	86.96	95.65	78.26	91.30	91.30	95.65	82.61	91.30	95.65	95.65	95.65	91.30	91.30	95.65	100	100	95.65	91.30	100	100	100	100	100	100	100	100	100
		WikiNews (FT/skip-gram)									FinText (FT/skip-gram)									FinText (FT/CBOW)								
MSE	Avg	-122.64	-113.51	-113.73	-116.13	-111.83	-112.19	-112.34	-110.49	-110.02	-157.90	-151.54	-148.47	-161.52	-149.48	-151.02	-147.88	-149.88	-145.44	-134.70	-119.37	-117.61	-137.10	-125.98	-99.99	-102.49	-121.60	-80.85
	Med	-52.25	-43.33	-44.83	-45.50	-44.76	-45.19	-46.00	-46.67	-45.70	-62.29	-58.74	-57.31	-60.09	-58.09	-61.11	-49.91	-54.01	-58.18	-21.60	-19.44	-13.91	-16.23	-18.40	-16.20	-18.75	-15.83	-25.58
RC	0.05	100	100	100	100	100	100	100	100	100	100	100	100	100	95.65	95.65	95.65	95.65	100	91.30	78.26	73.91	73.91	78.26	82.61	86.96	86.96	86.96
	0.10	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	95.65	82.61	86.96	82.61	95.65	91.30	91.30	91.30	86.96
QLIKE	Avg	-1.28	-1.24	-1.14	-1.23	-1.08	-1.21	-1.21	-1.20	-1.17	-1.23	-1.13	-1.20	-1.18	-1.12	-1.19	-1.12	-1.17	-1.21	0.02	-0.13	0.11	-0.28	-0.68	-0.67	-0.68	-0.76	-0.81
	Med	-1.34	-1.25	-1.14	-1.28	-1.11	-1.28	-1.29	-1.31	-1.25	-1.22	-1.25	-1.18	-1.21	-1.18	-0.89	-1.15	-1.25	-1.30	-0.29	-0.60	-0.37	-0.65	-0.48	-0.67	-1.04	-0.59	-0.87
RC	0.05	100	100	100	100	95.65	91.30	91.30	91.30	91.30	91.30	91.30	91.30	91.30	91.30	91.30	91.30	91.30	91.30	65.22	73.91	69.57	69.57	78.26	78.26	82.61	78.26	78.26
	0.10	100	100	100	100	100	100	100	100	100	95.65	95.65	95.65	95.65	95.65	95.65	100	95.65	95.65	65.22	78.26	73.91	91.30	86.96	82.61	86.96	86.96	91.30

Notes: ^a Average $\Delta_{MSE,j}$ and ^b Median $\Delta_{MSE,j}$ of 23 stocks, with a negative value indicates improvement, and a positive value indicates performance degradation. ^c Percentage of tickers with outstanding NLP-ML performance considering different number of CNN filters (3, 8, 13, 23, 28, 33, 38, and 43) at the 5% and 10% significance levels of the RC compared to the all HAR-family of models. ^d FastText algorithm.



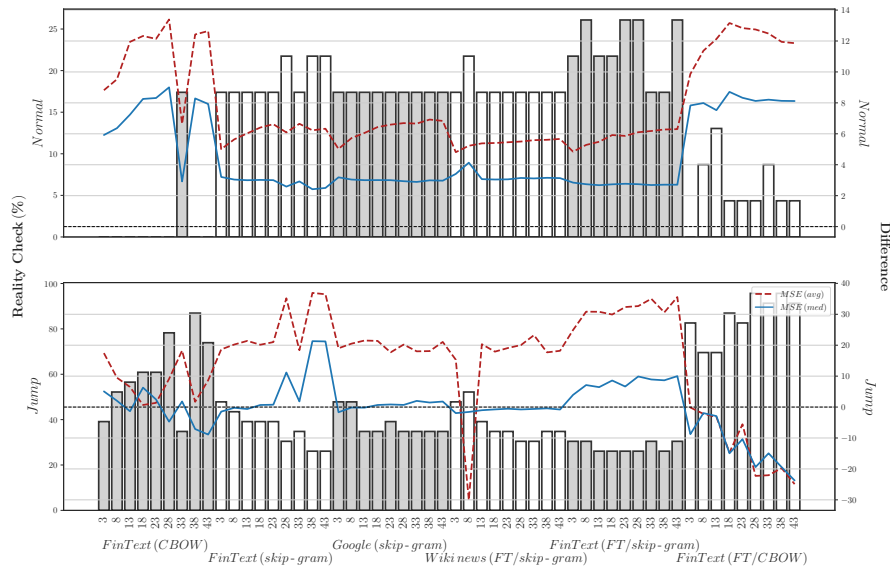
(a) Forecast evaluation under MSE



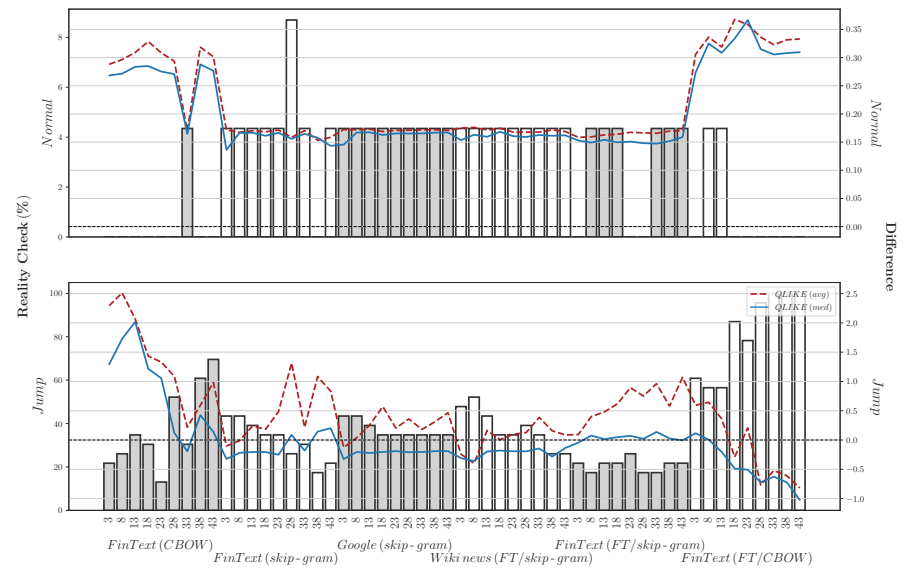
(b) Forecast evaluation under QLIKE

Figure 5: NLP-ML models forecasting performance - stock-related news

Notes: The bar chart is the percentage of tickers with the outstanding performance considering the MSE loss function in Figure 5a and QLIKE loss function in Figure 5b at the 0.05 significance level of the RC compared to the all HAR-family of models as the benchmark for every specified NLP-ML model. The values for the bar chart can be read from the left-hand axis. The dashed (solid) line represents the difference between the average (median) of the out-of-sample MSE(QLIKE)s of the specified NLP-ML model with the CHAR model (the best performing HAR-family model in Rahimikia and Poon (2020a)) for 23 tickers (negative value shows improvement, and positive value shows degradation in performance). The values for the dashed and solid lines can be read from the right-hand axis. The horizontal dashed line represents no improvement.



(a) Forecast evaluation under MSE



(b) Forecast evaluation under QLIKE

Figure 6: NLP-ML model performance - hot political news

Notes: The bar chart is the percentage of tickers with the outstanding performance considering the MSE loss function in Figure 6a and QLIKE loss function in Figure 6b at the 0.05 significance level of the RC compared to the all HAR-family of models as the benchmark for every specified NLP-ML model. The values for the bar chart can be read from the left-hand axis. The dashed (solid) line represents the difference between the average (median) of the out-of-sample MSE(QLIKE)s of the specified NLP-ML model with the CHAR model (the best performing HAR-family model in Rahimikia and Poon (2020a)) for 23 tickers (negative value shows improvement, and positive value shows degradation in performance). The values for the dashed and solid lines can be read from the right-hand axis. The horizontal dashed line represents no improvement.

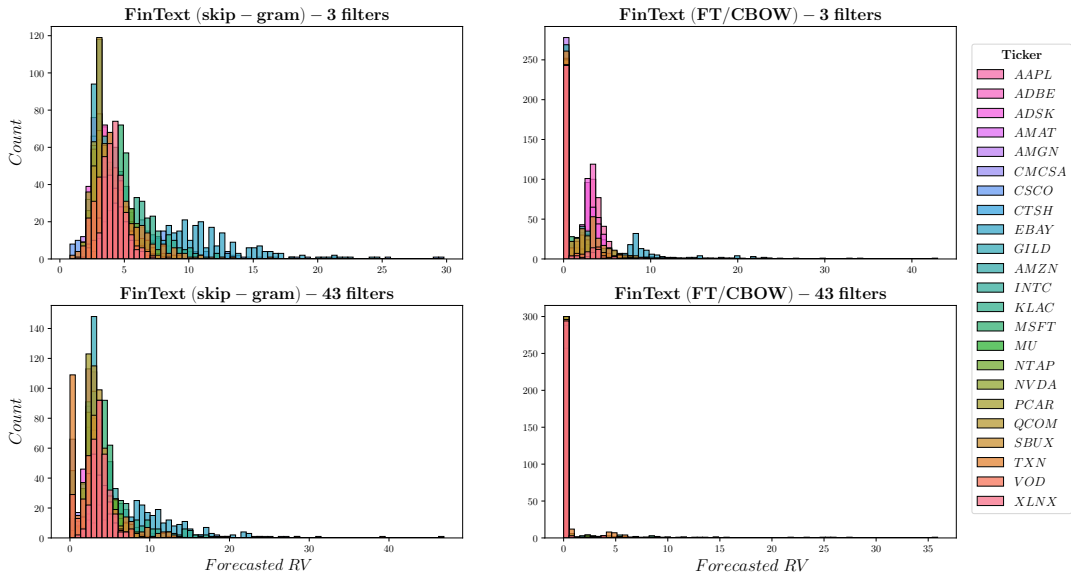


Figure 7: Distribution of RV forecasts - hot political news

Notes: Each figure shows the distribution of forecasted RV for all 23 analysed stocks regarding the specified NLP-ML model and the number of CNN filters.

in directional forecasts specially for normal volatility days.

5.2 Hot Political News

This subsection attempts to provide a performance summary of the hot political news for RV forecasting. The results for hot political news are presented in Figure 6. From this figure, in line with stock-related news in Subsection 5.1, NLP-ML models based on hot political news show improvement in forecasting power of volatility jumps but not in the same degree for normal volatility days. A comparison of stock-related news in Figure 5 and hot political news in Figure 6 also reveals that hot political news have less RV forecasting power for volatility jumps compared with stock-related news. However, focusing on RC results, this statistically significant improvement of hot political news is not negligible for both MSE and QLIKE loss functions. Interestingly, MDA results in Figure A2 show improvement in RV forecasting of normal volatility days for averagely more than 60% of tickers. These results draw our attention to the importance of considering hot political news (or other types of news) as a potential contributor to RV.

If we now turn to a detailed analysis of hot political news results, it can be seen in Figure 6 that the ‘FinText(FT/CBOW)’ group of MLP-ML models show substantially better forecasting performance for volatility jumps considering both MSE and QLIKE loss functions. In line with this finding, MDA results in Figure A2 reveal a generally similar pattern for volatility jumps which is not consistent with the findings from stock-related news. Figure 7 displays the distribution of forecasted RV for ‘FinText(skip-gram)’ with 3 and 43 filters (left figures) and ‘FinText(FT/CBOW)’ with 3 and 43 filters (right figures). A comparison of left and right figures reveals that the forecasted RV distribution from ‘FinText(FT/CBOW)’ models with both 3 and 43 filters is severely right-skewed, and more specifically for the case of ‘FinText(FT/CBOW)’ with 43 filters, the majority of forecasted RVs are zero or near zero. A closer inspection of reported results demonstrates that the forecasted RV is the same value for most days, and it is not changing over time. Therefore, the mentioned contradictory results need

Table 6: CHARx, ML group, and NLP-ML structure comparison

Criterion	Model group		
	CHARx	ML group	NLP-ML
Variable type	news(LM ^a) + financial	news(LM) + financial	news
News type	headline + body	headline + body	headline
Historical information	last 23 days	last 23 days	last day
Training sequence	daily	daily	every 5 days

^a LM is the Loughran-McDonald dictionary approach (Loughran and McDonald, 2011).

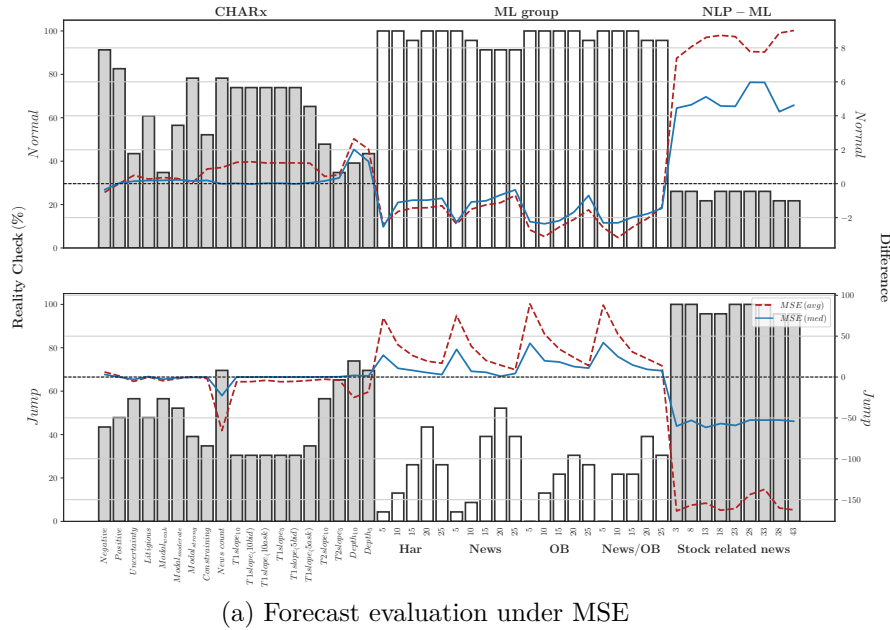
to be interpreted cautiously.

This subsection has attempted to provide a summary of other types of news like hot political and their contribution to RV forecasting. Understanding the impact of different types of news is vitally important but out of the scope of this study. Therefore, the following parts focus on stock-related news-based models as the best performing group of models in this study.

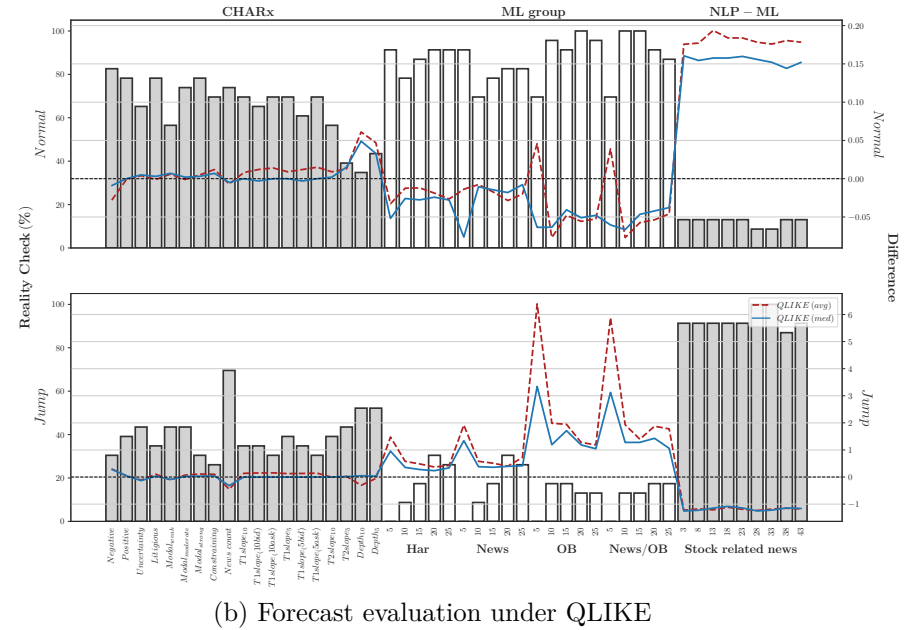
5.3 A Horse Race

Here, we compare three approaches for forecasting realised volatility, viz. CHARx, ML (with long short-term memory (LSTM)), and NLP-ML. Rahimikia and Poon (2020a) conducted a full investigation into the HAR-family of models and concluded that the CHAR structure is the most stable and provided the best RV forecasting performance. The authors went on to create CHARx, i.e., CHAR with exogenous variables and found some forecasting power from LOB variables and LM sentiment variables extracted from Dow Jones Newswires Text News Feed (stock-related news only). Later, Rahimikia and Poon (2020b) put all the CHARx and other variables in an LSTM model and found this ML structure significantly improved the forecasting power on normal volatility days. Here, we compare the forecasting power of CHARx and ML models (ML group) above with the best performing NLP-ML model ('FinText(skip-gram)') from the previous section. Table 6 shows the differences between these three approaches. From this table, it is clear that compared with CHARx and ML group, NLP-ML models incorporate substantially limited variable types (news not financial data), news types (headline, not body and headline together), historical information (one day, not 23 days), and training sequence (every five days not every day).

The results are presented in Figure 8. Regarding news sentiment, we have the LM dictionary sentiment variables (negative, positive, uncertainty, litigious, weak modal, moderate modal, strong modal, and constraining) and news count. The LOB variables are calculated based on 5 and 10 levels of bid, ask or both. The LOB variables include, among other variables, type-1 slope (Næs and Skjeltorp, 2006) and type-2 slope (Kalay et al., 2004) and depth. A few observations can be made from Figure 8. First, the ML group provided significantly greater forecasting power on normal volatility days but did poorly on volatility jump days. In contrast, NLP-ML provided significantly greater forecasting power on volatility jump days but did poorly on normal volatility days. The results are consistent for both MSE and QLIKE loss functions. Second, as discussed in Rahimikia and Poon (2020a), CHARx forecasting performance is less extreme, hovering around zero. 'News count', 'Depth5' and 'Depth10' could help forecast RV on volatility jump days. Negative sentiment



(a) Forecast evaluation under MSE



(b) Forecast evaluation under QLIKE

Figure 8: ML, extended CHARx, and NLP-ML models comparison

Notes: The bar chart is the percentage of tickers with the outstanding performance considering the MSE loss function in Figure 6a and QLIKE loss function in Figure 6b at the 0.05 significance level of the RC compared to the all HAR-family of models as the benchmark for every specified model. The values for the bar chart can be read from the left-hand axis. The dashed (solid) line represents the difference between the average (median) of the out-of-sample MSE(QLIKE)s of the specified model with the CHAR model (the best performing HAR-family model in Rahimikia and Poon (2020a)) for 23 tickers (negative value shows improvement, and positive value shows degradation in performance). The values for the dashed and solid lines can be read from the right-hand axis. The horizontal dashed line represents no improvement.

has marginal forecasting power on normal volatility days. In terms of directional forecasting results in Figure A3 presented in the Appendix, compared with the ML group, NLP-ML provides better forecasting performance for volatility jumps but worse for normal volatility days. Also, considering the RC results, in some cases, CHARx is marginally good for volatility jumps, and its forecasting performance is mixed on normal volatility days depending on whether one is using median or mean measure.

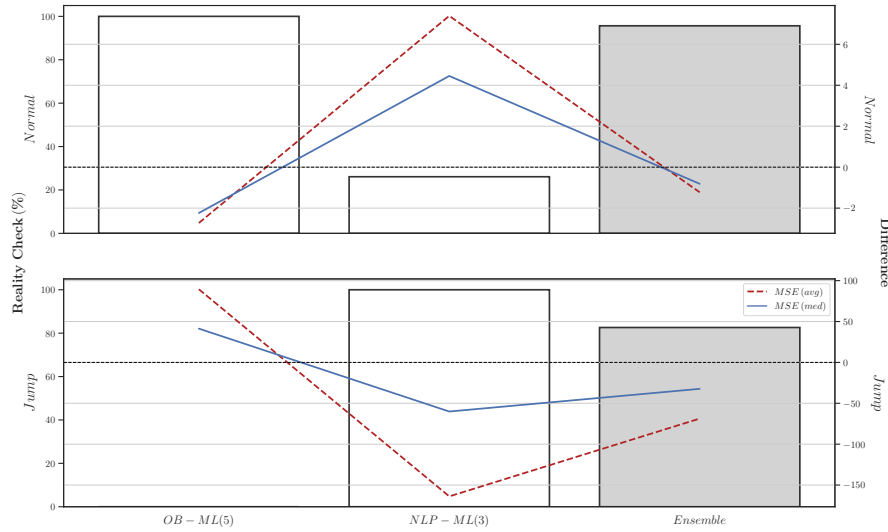
In summary, we deduce from the analyses here that, first, there is important textual information that cannot be extracted from the dictionary approach. Second, there is essential information in the financial numbers that is not in the text. Hence, in Subsection 5.4 below, we use an ensemble model to combine both sources of information to RV forecasting that works for both normal volatility days and volatility jump days.

5.4 Ensemble Model

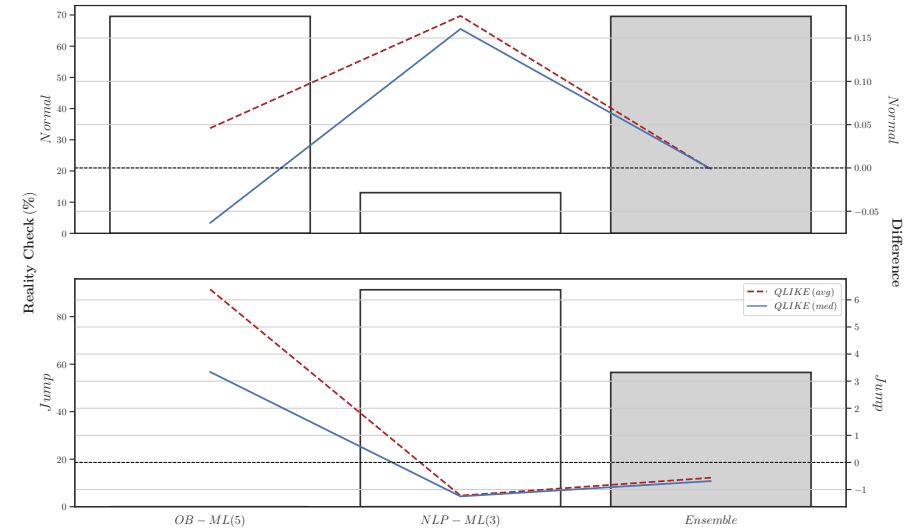
In this study, the ensemble model forecast is simply the arithmetic mean of the forecast from the best performing NLP-ML model from the previous section and the forecast from the best performing OB-ML model in Rahimikia and Poon (2020b). The OB-ML model contains 134 variables extracted from the LOB and autoregressive volatility variables in the HAR-family of models. Rahimikia and Poon (2020b) found the news sentiment variables to have only marginal predictive power. Also, because the NLP-ML model has a much more powerful textual information extraction routine, we have decided not to choose the news sentiment-based models from this study.

Figure 9 compares the forecasting performance of OB-ML with five units (following Rahimikia and Poon (2020b)), NLP-ML ('FinText(skip-gram)') with three CNN filters, and the ensemble model. The figure shows that the ensemble model retains the favourable forecasting power from the two sub-components, producing significant forecasting power for jump days (comparable to NLP-ML) and on normal volatility days (comparable to OB-ML). On normal volatility days, the ensemble model dominates the HAR-family of models for 95.65% and 82.61% of tickers, with MSE and QLIKE loss functions, respectively. On volatility jump days, the ensemble model dominates the HAR-family of models for 69.57% and 56.52% of tickers, with MSE and QLIKE loss functions, respectively. Further analysis of the MDA loss function results in Figure A4 generally confirms these findings. On normal volatility days, the ensemble model retains some forecasting power from OB-ML, outperforming the HAR-family of models at the 5% significance level for 91.30% of tickers, and not well but still better than OB-ML on volatility jump days.

So far, we have a fixed number of units for OB-ML and a fixed number of CNN filters for NLP-ML. Figure 10 shows the results for a wide range of parameter choices in a heatmap. Specifically, the x-axis is for 3, 8, 13, 18, 23, 28, 33, 38, and 43 CNN filters in NLP-ML, and the y-axis is for 5, 10, 15, 20, and 25 as the number of units in the OB-ML model. The top (bottom) row is for normal volatility days (jumps). MSE (QLIKE)-Avg (Med) is the difference between the average (median) of the out-of-sample MSE (QLIKE)s forecasts for 23 tickers of the specified ensemble model with CHAR as the benchmark. MSE (QLIKE)-RC is the percentage of tickers with the outstanding performance against all HAR-family of models with MSE (QLIKE) loss function at the 5% significance level. More



(a) Forecast evaluation under MSE



(b) Forecast evaluation under QLIKE

Figure 9: Performance of ensemble model, against OB-ML(5) and NLP-ML(3)

Notes: The bar chart is the percentage of tickers with the outstanding performance considering the MSE loss function in Figure 9a and QLIKE loss function in Figure 9b at the 0.05 significance level of the RC compared to the all HAR-family of models as the benchmark for OB-ML model with five units as the chosen model from Rahimikia and Poon (2020b), NLP-ML model with three filters as one of the best performing models in this study, and the ensemble model. The values for the bar chart can be read from the left-hand axis. The dashed (solid) line represents the difference between the average (median) of the out-of-sample MSE(QLIKE)s of the specified model with the CHAR model (the best performing HAR-family model in Rahimikia and Poon (2020a)) for 23 tickers (negative value shows improvement, and positive value shows degradation in performance). The values for the dashed and solid lines can be read from the right-hand axis. The horizontal dashed line represents no improvement.

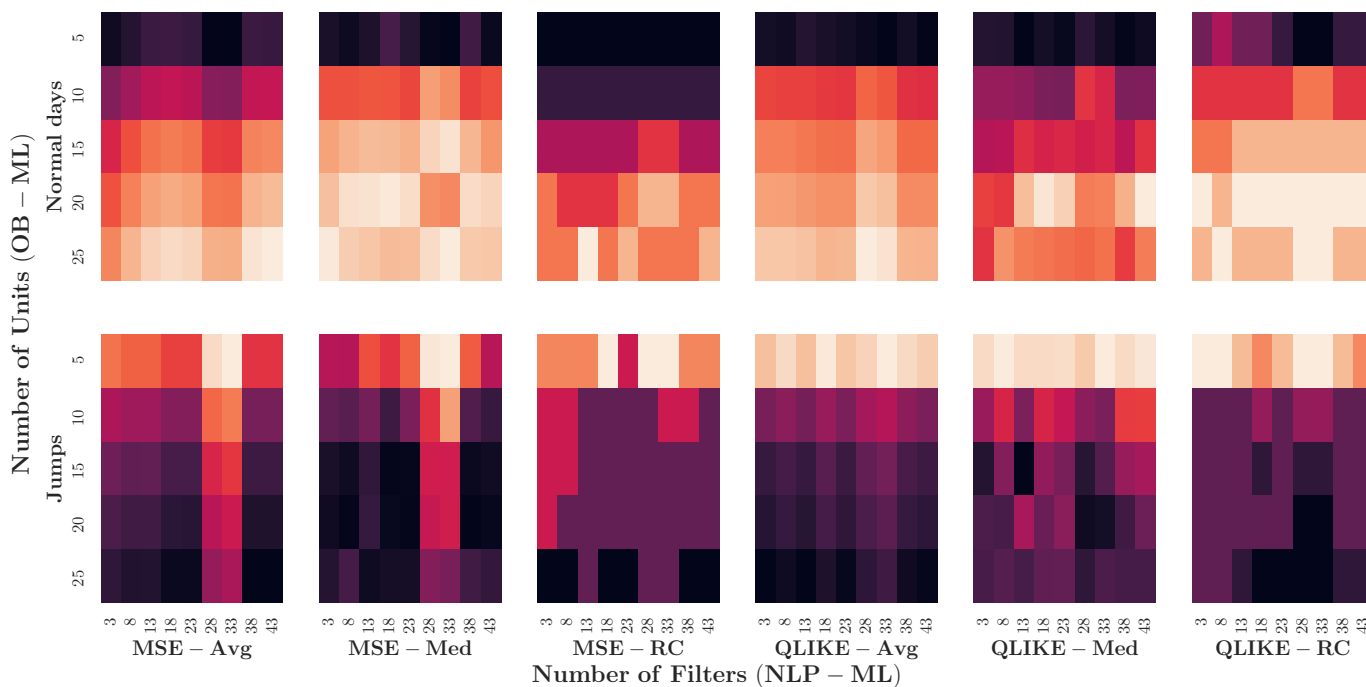


Figure 10: Ensemble models comparison - Number of Units vs. Filters (MSE & QLILE)

Notes: The x-axis is the NLP-ML model with 3, 8, 13, 18, 23, 28, 33, 38, and 43 as the number of filters, and the y-axis is the OB-ML model in Rahimikia and Poon (2020b) with 5, 10, 15, 20, and 25 as the number of units. The first(second) row contains the results for normal volatility days(jumps). MSE(QLIKE)-Avg(Med) is the difference between the average(median) of the out-of-sample MSE(QLIKE)s of the specified NLP-ML model in the x-axis and OB-ML in the y-axis with the CHAR model (the best performing HAR-family model in Rahimikia and Poon (2020a)) for 23 tickers. MSE(QLIKE)-RC is the percentage of tickers with the outstanding performance considering the MSE(QLIKE) loss function at the 0.05 significance level of the RC compared to the all HAR-family of models as the benchmark for every specified NLP-ML model in the x-axis and OB-ML in the y-axis. Darker cell shows improvement, and lighter cell shows degradation in performance.

importantly, the darker cell shows improvement, and the lighter cell shows degradation in performance. Hence, Figure 10 shows the performance is not sensitive to the number of CNN filters in the NLP-ML model. This means one could choose a small number to save time. However, the performance is sensitive to the number of units in OB-ML. A smaller number is preferred for normal volatility days, and a larger number is preferred for volatility jump days. This finding is in line with the findings in Rahimikia and Poon (2020b). The heatmap for the MDA loss function results in Figure A5 reached the same conclusion.

In summary, this ensemble approach stresses the importance of the information content of both financial numbers and news for forecasting RV again. When we study news with ML, a simpler structure with a small number of CNN filters is sufficient, and this textual information tends to be good for predicting volatility jumps. However, with financial numbers, the process is different, whether it is for a normal day forecast or a jump day forecast. A compromise is choosing a small number of units for normal day forecasts without damaging the jump day forecast and letting the news and NLP-ML model take care of jump day forecasts. Future research could investigate if weighted average or even more complex ensemble models could perform better.

5.5 Explainable AI Results

Over the last few years, great efforts have been made to understand ML models better, which are often described as black-box. Here, we will use two Explainable AI methods, viz. IG and SHAP, to analyse the impact of textual information in volatility forecasting in an ML model. A brief review of IG and SHAP is in Subsection 4.3.

5.5.1 ‘Loss’ in Stock-Related News

Figure 11 presents the IG (on the left) and SHAP (on the right) values in the out-of-sample period for the token ‘loss’ produced from ‘FinText(skip-gram)’ NLP-ML model with three as the number of CNN filters, the best performing NLP-ML model from the previous section. For both IG and SHAP, a larger positive (negative) value means a larger increase (decrease) in the RV forecast. For instance, considering SHAP values for the first ticker (AAPL), all appearances of the token ‘loss’ increased the RV forecast but with slightly different magnitudes. The behaviour is very different for the second ticker (ADSK); apart from one incident, all the other appearances of ‘loss’ reduced the RV forecasts.

In Figure 11 it is seen that ‘loss’ led to increase (decrease) in RV forecasts 42 (18) times. This is the case whether we measure the impact with IG or with SHAP. ‘Loss’ is an important token for ‘MU’ but has almost no effect on ‘AMGN’. In contrast, the dictionary approach will classify ‘loss’ always as a negative term and possibly with a prediction that it always increases RV forecast by a fixed amount. The ML via *FinText* word embedding allows for a greater variety of contextual relationships between ‘loss’ and RV forecast.

5.5.2 ‘Donald Trump’ in Hot Political News

Here, ‘donald_trump’ is chosen as a political bigram (i.e. a pair of words). Figure 12 presents the IG (on the left) and SHAP (on the right) values in the out-of-sample period of hot political news for this

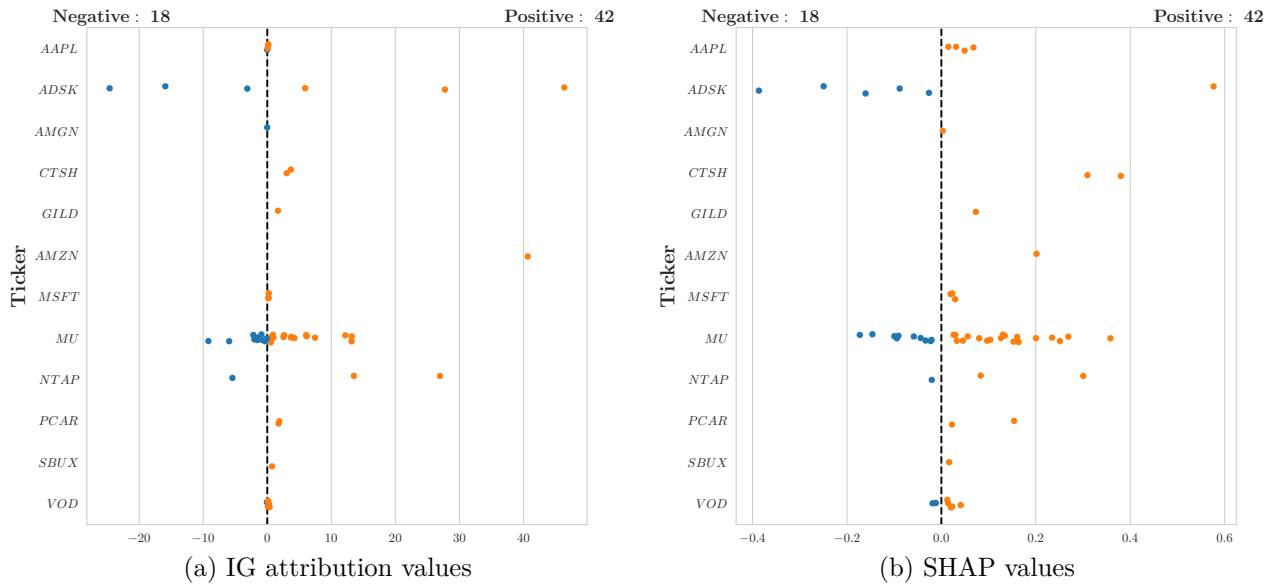


Figure 11: Explainer results for ‘loss’ (ticker-related news)

Notes: The left (right) figure presents the IG attribution (SHAP) values for ‘loss’ as the chosen token considering the stock-related news published during the out-of-sample period. The x-axis is the reported explainer value, and the y-axis is the ticker name.

bigram produced from ‘FinText(skip-gram)’ with 3 CNN filters.

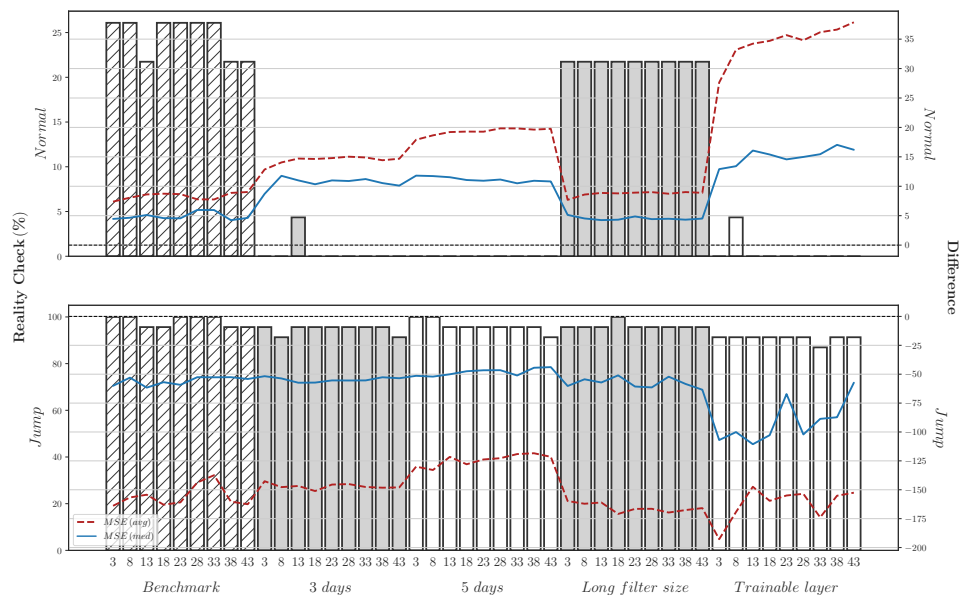
From the IG measure, there are now 1900 (492) cases when ‘donald_trump’ led to an increase (decrease) in RV forecast. For the SHAP measure, this number is 1589 (803). So ‘donald_trump’ is a highly sensitive bigram and tends to increase market volatility. What is interesting about the results from hot political news is that this approach is able to identify the impact of any desired term, not just a limited group of terms in a dictionary.

5.5.3 Visualisation for Apple

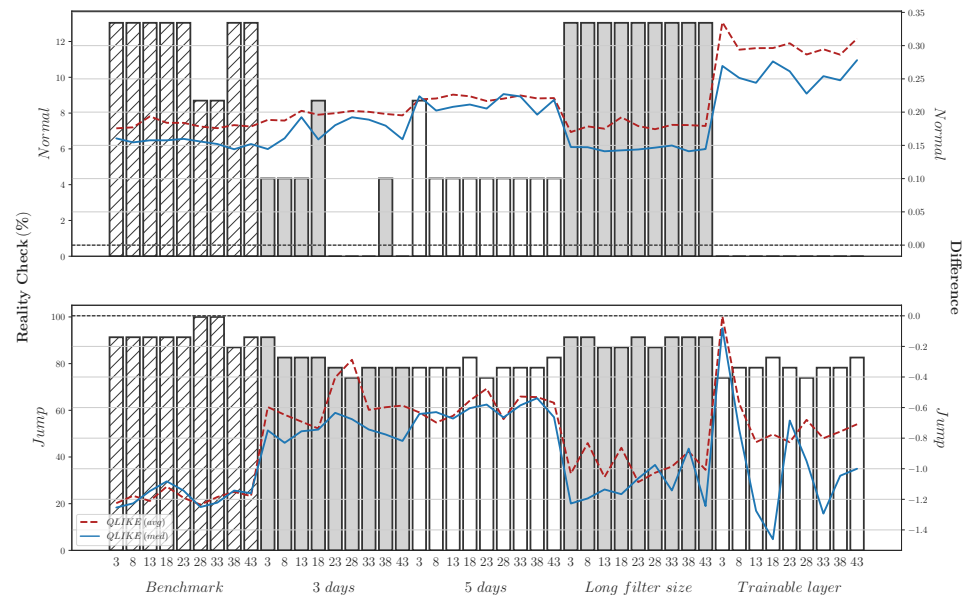
To gain some appreciation on the context within which the token affects volatility forecasts, we present a visualisation using the stock-related news for Apple (with ticker ‘AAPL’) on 2016-10-2 and hot political news on 2016-07-20. Figure 13 presents SHAP visualisation in the top panel for stock-related news about Apple and IG visualisation in the bottom panel for hot political news. Note that ‘NONE’ is used repeatedly in the bottom panel as a padding procedure. The token coloured red (blue) indicates an increase (decrease) in the RV forecast. A darker colour indicates a greater intensity.

Figure 13a shows that the majority of the highlighted tokens are related to the fourth-quarter earnings of Apple, and these tokens together increased the RV forecasts. Not only that the NLP-ML model learned the link between the earnings release and a higher volatility forecast on the following day, but also it appears to have also identified the news tag, i.e. ‘> aapl’, that might have been used to denote earning related information. For hot political news in Figure 13b, the key phrases, ‘republican national convention nominates donald trump’, ‘donald trump secures republican’, and ‘gop¹ presidential nomination’ caused an increase in the RV forecast. This illustrates the capability of NLP-ML in detecting important political news that has an effect on volatility. Moreover, the different

¹GOP is short for ‘Grand Old Party’, a colloquium for the Republican party.



(a) Forecast evaluation under MSE



(b) Forecast evaluation under QLIKE

Figure 14: Robustness checks (performance comparisons)

Notes: The bar chart is the percentage of tickers with the outstanding performance considering the MSE loss function in Figure 14a and QLIKE loss function in Figure 14b at the 0.05 significance level of the RC compared to the all HAR-family of models as the benchmark. Benchmark group (hashed bars) is ‘FinText(skip-gram)’ group in Figure 5. For ‘3 days’ and ‘5 days’, the time duration of input data is changed from one day to three and five, respectively. For the ‘Long filter size’ group, 8, 9 and 10 are chosen as filter size (kernel size) instead of 3, 4, and 5. Furthermore, regarding the ‘Trainable layer’ group, the embedding layer of the NLP-ML model is set to trainable. The values for the bar chart can be read from the left-hand axis. The dashed (solid) line represents the difference between the average (median) of the out-of-sample MSE(QLIKE)s of the specified model with the CHAR model (the best performing HAR-family model in Rahimikia and Poon (2020a)) for 23 tickers (negative value shows improvement, and positive value shows degradation in performance). The values for the dashed and solid lines can be read from the right-hand axis. The horizontal dashed line represents no improvement.

shades of colour show that NLP-ML assigns very different weights to the words in the phrase reflecting its ability to handle complexity.

6 Robustness Checks

In this section, we assess the impact of NLP-ML parameters and forecasting structures on the RV forecasting performance we have reported so far. Here, ‘FinText(skip-gram)’ is the benchmark, our best performing NLP-ML model with 3, 8, 13, 18, 23, 28, 33, 38, and 43 CNN filters. The robustness check studies four modifications to the benchmark. First, the ‘3 days’ group changes the input from news headlines of the previous day to the previous three days. Similarly, the second modification takes as input the headlines from the previous 5 days.¹ The third modification, ‘Long filter size’, changes the filter (kernel) size from 3, 4, and 5 to 8, 9 and 10. Finally, the fourth modification, ‘Trainable layer’, makes the fixed embedding layer trainable. This means that the initial embedding layer parameters obtained from the *FinText* could be changed. The results are presented in Figure 14.

Figure 14 reveals that moving to ‘3 days’ and ‘5 days’ input duration causes substantial degradation in forecasting performance of normal volatility days; apart from an increase in mean and median values, the RC values substantially reduced or dropped to zero. A degradation in forecasting performance on volatility jump days is also observable, especially when QLIKE is the loss function. The directional forecast results for the MDA loss function in Figure A6 show a degradation in the forecasting performance, especially on volatility jump days. These findings may be due to the fact that long input duration makes the NLP-ML model more complicated and would require more than 2046 days of data for training. This could be one major challenge researchers may face in future studies in this area.

Figure 14 also shows little effect from changing the filter size. This finding is generally consistent with the MDA results in Figure A6. Finally, making the word embedding layer trainable causes a substantial degradation in RV forecasting performance of normal volatility days while not providing any noticeable improvement on volatility jump days. The MDA results in Figure A6 do not show any substantial difference in forecasting performance. The main reason why the NLP-ML models with trainable embedding layers did not improve forecasting performance lies in the fact that this change substantially increases the number of trainable parameters. Therefore, more training samples are required, which is again not feasible with the current financial data sample size.

7 Conclusions

In this paper, we developed a financial word embedding called *FinText*², based on Dow Jones Newswires Text News Feed database (2000-2015) using Word2Vec and FastText algorithms. Our financial word embedding performed less well on general-purpose benchmarks when compared with Google’s and Facebook’s word embeddings. However, when challenged with detecting unique financial relation-

¹Both modifications will reduce the times when there is no news. For the 23 tickers, moving to three and five days decreases the number of days without news by 45.22% and 64.32%, respectively, for in-sample data, and 49.34% and 69.89%, respectively, for out-of-sample data.

²*FinText* word embeddings are available for download from rahimikia.com/fintext.

ships, *FinText* is better and more sensitive in detecting financial jargon. However, our focus in this paper is to test these pre-trained word embeddings in realised volatility forecasting using a machine learning model. Using data for 23 NASDAQ stocks from 27 July 2007 to 18 November 2016, we found evidence that headlines of stock-related news produced a substantial improvement on forecasting daily realised volatility on volatility jump days, beating all HAR-family of models in Rahimikia and Poon (2020a) and the machine learning models in Rahimikia and Poon (2020b) that did not make use of word embeddings.

For forecasting realised volatility on volatility jump days using stock-related news, our proposed model, named NLP-ML, performs marginally better than general word embeddings such as Google and Facebook word embeddings. This study has also identified that political news, to a lesser extent, is crucial for improving the realised volatility forecasting performance. Furthermore, since financial news with NLP-ML model performs well on volatility jump days, while market limit order book LOB-ML model performs well on normal volatility days in Rahimikia and Poon (2020b), we create an ensemble model, combining both NLP-ML and LOB-ML, which dominates all HAR-family of models on both normal volatility days and days with volatility jumps. Our ensemble model is simply the arithmetic average of forecasts obtained from the best performing NLP-ML and LOB-ML. Finally, we use two Explainable AI methods to measure the impact of given tokens on realised volatility forecasts. Such performance attribution is not feasible in the classical dictionary-based approach in textual analysis.

In conclusion, we demonstrate that our purpose-built financial word embeddings have superior volatility forecast power when used to analyse financial news headlines via a machine learning model. Here, we forecast realised volatility of 23 NASDAQ stocks; it will be interesting to extend the forecasting exercise to the other financial contexts, such as stock returns, limit order book depth, trading volume etc. Furthermore, the scope of this study was limited because the training sample is small. As a result, we tested only news headlines but not the news bodies. Moreover, instead of daily forecasting, it would be interesting to test the NLP-ML forecasting power for higher-frequency data. We also hope that this study paves the way for other state-of-the-art natural language processing and machine learning research in different financial contexts beyond realised volatility forecasting.

References

- Adämmer, P. and R. A. Schüssler (2020). Forecasting the equity premium: mind the news! *Review of Finance* 24(6), 1313–1355.
- Agirre, E., E. Alfonseca, K. Hall, J. Kravalova, M. Pasca, and A. Soroa (2009). A study on similarity and relatedness using distributional and wordnet-based approaches.
- Ban, G.-Y., N. El Karoui, and A. E. Lim (2018). Machine learning and portfolio optimization. *Management Science* 64(3), 1136–1154.
- Bianchi, D., M. Büchner, and A. Tamoni (2021). Bond risk premiums with machine learning. *The Review of Financial Studies* 34(2), 1046–1089.
- Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5, 135–146.
- Bollerslev, T., A. J. Patton, and R. Quaedvlieg (2016). Exploiting the errors: A simple approach for improved volatility forecasting. *Journal of Econometrics* 192(1), 1–18.
- Bubna, A., S. R. Das, and N. Prabhala (2020). Venture capital communities. *Journal of Financial and Quantitative Analysis* 55(2), 621–651.
- Bybee, L., B. T. Kelly, A. Manela, and D. Xiu (2020). The structure of economic news. Technical report, National Bureau of Economic Research.
- Chen, L., M. Pelger, and J. Zhu (2020). Deep learning in asset pricing. *Available at SSRN 3350138*.
- Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa (2011). Natural language processing (almost) from scratch. *Journal of machine learning research* 12(ARTICLE), 2493–2537.
- Conrad, C. and R. F. Engle (2021). Modelling volatility cycles: the $(mf)^2$ garch model. *Available at SSRN*.
- Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics* 7(2), 174–196.
- Corsi, F. and R. Reno (2009). Har volatility modelling with heterogeneous leverage and jumps. *Available at SSRN 1316953*.
- Engle, R. F. and S. Martins (2020). Measuring and hedging geopolitical risk.
- Engle, R. F. and V. K. Ng (1993). Measuring and testing the impact of news on volatility. *The journal of finance* 48(5), 1749–1778.
- Gentzkow, M., B. Kelly, and M. Taddy (2019). Text as data. *Journal of Economic Literature* 57(3), 535–74.
- Gu, S., B. Kelly, and D. Xiu (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies* 33(5), 2223–2273.
- Gu, S., B. Kelly, and D. Xiu (2021). Autoencoder asset pricing models. *Journal of Econometrics* 222(1), 429–450.
- Hill, F., R. Reichart, and A. Korhonen (2015). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* 41(4), 665–695.
- Jiang, J., B. T. Kelly, and D. Xiu (2020). (re-) imag (in) ing price trends. *Chicago Booth Research Paper* (21-01).
- Joulin, A., E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov (2016). Fasttext.zip: Compressing text classification models.

- Kalay, A., O. Sade, and A. Wohl (2004). Measuring stock illiquidity: An investigation of the demand and supply schedules at the tase. *Journal of Financial Economics* 74(3), 461–486.
- Ke, Z. T., B. T. Kelly, and D. Xiu (2019). Predicting returns with text data. Technical report, National Bureau of Economic Research.
- Kim, Y. (2014). Convolutional neural networks for sentence classification. *CoRR abs/1408.5882*.
- Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Li, K., X. Liu, F. Mai, and T. Zhang (2020). The role of corporate culture in bad times: Evidence from the covid-19 pandemic. *Journal of Financial and Quantitative Analysis*, 1–68.
- Lim, B., S. Zohren, and S. Roberts (2019). Enhancing time-series momentum strategies using deep neural networks. *The Journal of Financial Data Science* 1(4), 19–38.
- Loughran, T. and B. McDonald (2011). When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance* 66(1), 35–65.
- Loughran, T. and B. McDonald (2016). Textual analysis in accounting and finance: A survey. *Journal of Accounting Research* 54(4), 1187–1230.
- Lundberg, S. and S.-I. Lee (2017). A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.
- Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Mikolov, T., E. Grave, P. Bojanowski, C. Puhersch, and A. Joulin (2018). Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.
- Mikolov, T., I. Sutskever, K. Chen, G. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.
- Morin, F. and Y. Bengio (2005). Hierarchical probabilistic neural network language model. In *Aistats*, Volume 5, pp. 246–252. Citeseer.
- Næs, R. and J. A. Skjeltorp (2006). Order book characteristics and the volume–volatility relation: Empirical evidence from a limit order market. *Journal of Financial Markets* 9(4), 408–432.
- Obaid, K. and K. Pukthuanthong (2021). A picture is worth a thousand words: Measuring investor sentiment by combining machine learning and photos from news. *Journal of Financial Economics*.
- Patton, A. J. (2011). Volatility forecast comparison using imperfect volatility proxies. *Journal of Econometrics* 160(1), 246–256.
- Patton, A. J. and K. Sheppard (2015). Good volatility, bad volatility: Signed jumps and the persistence of volatility. *Review of Economics and Statistics* 97(3), 683–697.
- Poh, D., B. Lim, S. Zohren, and S. Roberts (2021). Building cross-sectional systematic strategies by learning to rank. *The Journal of Financial Data Science* 3(2), 70–86.
- Politis, D. N. and J. P. Romano (1994). The stationary bootstrap. *Journal of the American Statistical association* 89(428), 1303–1313.
- Rahimikia, E. and S.-H. Poon (2020a). Big data approach to realised volatility forecasting using har model augmented with limit order book and news. *Available at SSRN 3684040*.
- Rahimikia, E. and S.-H. Poon (2020b). Machine learning for realised volatility forecasting. *Available at SSRN 3707796*.

- Shapiro, A. H., M. Sudhof, and D. J. Wilson (2020). Measuring news sentiment. *Journal of Econometrics*.
- Shrikumar, A., P. Greenside, and A. Kundaje (2017). Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pp. 3145–3153. PMLR.
- Sirignano, J. and R. Cont (2019). Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance* 19(9), 1449–1459.
- Sundararajan, M., A. Taly, and Q. Yan (2017). Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR.
- Wood, K., S. Roberts, and S. Zohren (2021). Slow momentum with fast reversion: A trading strategy using deep learning and changepoint detection. *arXiv preprint arXiv:2105.13727*.
- Wu, W., J. Chen, Z. Yang, and M. L. Tindall (2020). A cross-sectional machine learning approach for hedge fund return prediction and selection. *Management Science*.
- Zhang, Z. and S. Zohren (2021). Multi-horizon forecasting for limit order books: Novel deep learning approaches and hardware acceleration using intelligent processing units. *arXiv preprint arXiv:2105.10430*.
- Zhang, Z., S. Zohren, and S. Roberts (2018). Bdlb: Bayesian deep convolutional neural networks for limit order books. *arXiv preprint arXiv:1811.10041*.
- Zhang, Z., S. Zohren, and S. Roberts (2020). Deep reinforcement learning for trading. *The Journal of Financial Data Science* 2(2), 25–40.

Table A1: Textual data cleaning rules

Primary	
<p>Extracting body of news from XML Converting XML to text (parsing) Removing tables</p>	<p>Removing XML-Encoding Characters (XMLENCOD) Converting uppercase letters to lowercase letters</p>
Begins with	
<p>(END) XX for (more further) (information from marketwatch), please visit: XX (EMAIL; @XX) copyright XXXX, XX URL (more to follow) XX end of (message corporate news) XX source: XX URL XX contributed to this article XX view source version on XX (=—————) view original content with multimedia XX readers can alert XX view original content: XX media inquiries: XX readers: send feedback to XX follow us on XX contact(s): XX please refer to URL XX find out more at URL XX XX enquiries: XX -by XX, dow jones newswires XX</p>	<p>(email e-mail): XX (phone fax contact dgap-ad-hoc dgap-news): XX image available: XX source: XX to read more, visit: XX (view source view original content) (with on) XX (investor relations investor contact) XX like us on XX (copyright (c) ©) XX XX can be found at URL XX by dow jones newswires XX (write to follow) XX at EMAIL (phone tel telephone mobile contact inquiries comment): XX (contact information media contact contact client services internet) XX click here to subscribe to XX to learn more about XX (website web site): URL XX contact us in XX to receive news releases by (e-mail email) XX full story at XX</p>
Ends with	
<p>(more to follow) (fax tell contact dgap-ad-hoc dgap-news): ratings actions from baystreet: cannot parse story lipper indexes:to subscribe to for full details, please click on</p>	<p>view original content XX: (contacts web site): (= - - _ · -) for notes, kindly refer following is the related link:</p>
General	
<p>(linkedin facebook fb): XX (twitter ig): XX (attachment attachments): XX please visit XX follow us on XX All rights reserved</p>	<p>(URL (and &) XX) (EMAIL (and &) EMAIL) this information was brought to you by XX write to EMAIL to receive our XX URL more at, XX URL</p>
Final checks	
<p>Removing links and emails Removing both the leading and the trailing space(s)</p>	<p>Removing short news (lower than 25 characters) Removing phone numbers</p>

Table A2: Out-of-sample forecasting performance (MDA) and RC results for NLP-ML models - stock-related news

Normal days	Units	FinText (CBOW)									FinText (skip-gram)									Google (skip-gram)								
		3	8	13	18	23	28	33	38	43	3	8	13	18	23	28	33	38	43	3	8	13	18	23	28	33	38	43
MDA	Avg	0.50 ^a	0.64	0.73	0.32	0.05	-0.08	0.13	0.15	0.18	0.14	0.11	0.24	0.27	0.30	0.08	0.08	0.43	0.56	-0.04	-0.07	-0.10	0.22	0.06	-0.04	0.17	0.11	0.30
	Med	0.17 ^b	0.89	0.91	1.30	0.14	-0.73	-0.36	0.52	-0.20	-0.20	0.57	0.37	0.14	-0.22	-0.21	-0.21	0.00	0.55	-0.58	-0.58	-0.21	-0.21	-0.21	-0.21	-0.21	-0.58	0.14
RC	0.05	69.57 ^c	69.57	69.57	69.57	56.52	65.22	69.57	60.87	65.22	60.87	60.87	60.87	65.22	65.22	65.22	65.22	65.22	73.91	56.52	56.52	56.52	60.87	65.22	65.22	65.22	65.22	65.22
	0.10	73.91 ^c	78.26	73.91	73.91	69.57	65.22	69.57	65.22	65.22	65.22	69.57	69.57	73.91	73.91	69.57	69.57	78.26	78.26	65.22	65.22	69.57	73.91	69.57	69.57	65.22	65.22	69.57
		WikiNews (FT ^d /skip-gram)									FinText (FT/skip-gram)									FinText (FT/CBOW)								
MDA	Avg	-0.76	-0.72	-0.58	-0.64	-0.42	-0.53	-0.56	-0.61	-0.50	0.06	0.24	0.18	0.32	0.16	0.14	0.21	0.42	0.39	0.21	-0.19	0.31	0.20	-0.36	-0.52	-1.03	-1.40	-1.42
	Med	-0.95	-0.59	-0.59	-0.59	-0.58	-0.37	-0.75	-0.59	-0.21	0.00	0.17	0.14	0.16	0.00	0.16	0.14	-0.20	0.75	0.55	-0.20	0.16	0.14	-0.94	-0.22	-1.45	-1.72	-1.72
RC	0.05	56.52	56.52	60.87	60.87	65.22	60.87	60.87	65.22	65.22	60.87	60.87	65.22	60.87	65.22	60.87	56.52	65.22	65.22	60.87	56.52	65.22	65.22	60.87	52.17	60.87	39.13	47.83
	0.10	60.87	65.22	65.22	65.22	69.57	65.22	69.57	65.22	69.57	60.87	65.22	69.57	69.57	69.57	65.22	69.57	69.57	69.57	69.57	69.57	65.22	65.22	69.57	69.57	60.87	56.52	47.83
Jumps		FinText (CBOW)									FinText (skip-gram)									Google (skip-gram)								
MDA	Avg	-9.61	-10.93	-11.70	-10.73	-9.96	-10.61	-8.55	-11.02	-9.69	-8.25	-7.96	-8.34	-8.21	-8.52	-7.87	-8.45	-8.10	-8.23	-6.69	-7.03	-7.48	-7.87	-7.36	-7.87	-8.31	-8.02	-8.17
	Med	-8.33	-9.38	-11.67	-11.11	-11.54	-9.09	-7.69	-12.20	-7.69	-6.25	-6.25	-8.33	-8.33	-6.25	-8.33	-7.69	-6.25	-5.88	-5.56	-6.25	-6.25	-7.69	-6.25	-6.67	-8.33	-6.25	-7.69
RC	0.05	26.09	8.70	13.04	21.74	30.44	13.04	17.39	13.04	21.74	43.48	34.78	43.48	39.13	39.13	39.13	30.44	47.83	43.48	43.48	43.48	43.48	34.78	43.48	34.78	30.44	34.78	30.44
	0.10	34.78	26.09	34.78	34.78	43.48	30.44	39.13	30.44	34.78	56.52	47.83	52.17	52.17	52.17	65.22	56.52	52.17	52.17	60.87	60.87	65.22	60.87	60.87	56.52	60.87	56.52	60.87
		WikiNews (FT/skip-gram)									FinText (FT/skip-gram)									FinText (FT/CBOW)								
MDA	Avg	-5.83	-5.84	-5.90	-6.49	-6.83	-6.87	-7.04	-6.58	-7.30	-7.53	-8.28	-8.10	-8.15	-8.64	-7.93	-8.27	-8.19	-7.59	-11.66	-10.45	-10.26	-8.79	-10.09	-7.03	-8.31	-5.25	-5.10
	Med	-5.56	-5.56	-5.56	-6.06	-6.67	-7.69	-7.69	-8.33	-8.33	-5.88	-6.06	-6.25	-5.88	-6.25	-7.89	-6.25	-8.33	-6.25	-9.38	-11.76	-8.33	-8.82	-8.33	-7.67	-6.25	-3.13	-3.13
RC	0.05	43.48	39.13	47.83	34.78	39.13	34.78	39.13	43.48	39.13	47.83	34.78	39.13	34.78	34.78	39.13	43.48	43.48	39.13	13.04	30.44	26.09	17.39	26.09	21.74	21.74	26.09	39.13
	0.10	65.22	65.22	69.57	60.87	65.22	65.22	65.22	69.57	60.87	60.87	52.17	56.52	52.17	52.17	52.17	56.52	52.17	52.17	17.39	39.13	34.78	43.48	39.13	43.48	43.48	56.52	73.91

Notes: ^a Average $\Delta_{MDA,j}$ and ^b Median $\Delta_{MDA,j}$ of 23 stocks, with a negative value indicates degradation, and a positive value indicates performance improvement. ^c Percentage of tickers with outstanding NLP-ML performance considering different number of CNN filters (3, 8, 13, 23, 28, 33, 38, and 43) at the 5% and 10% significance levels of the RC compared to all HAR-family of models based on MDA. ^d FastText algorithm.

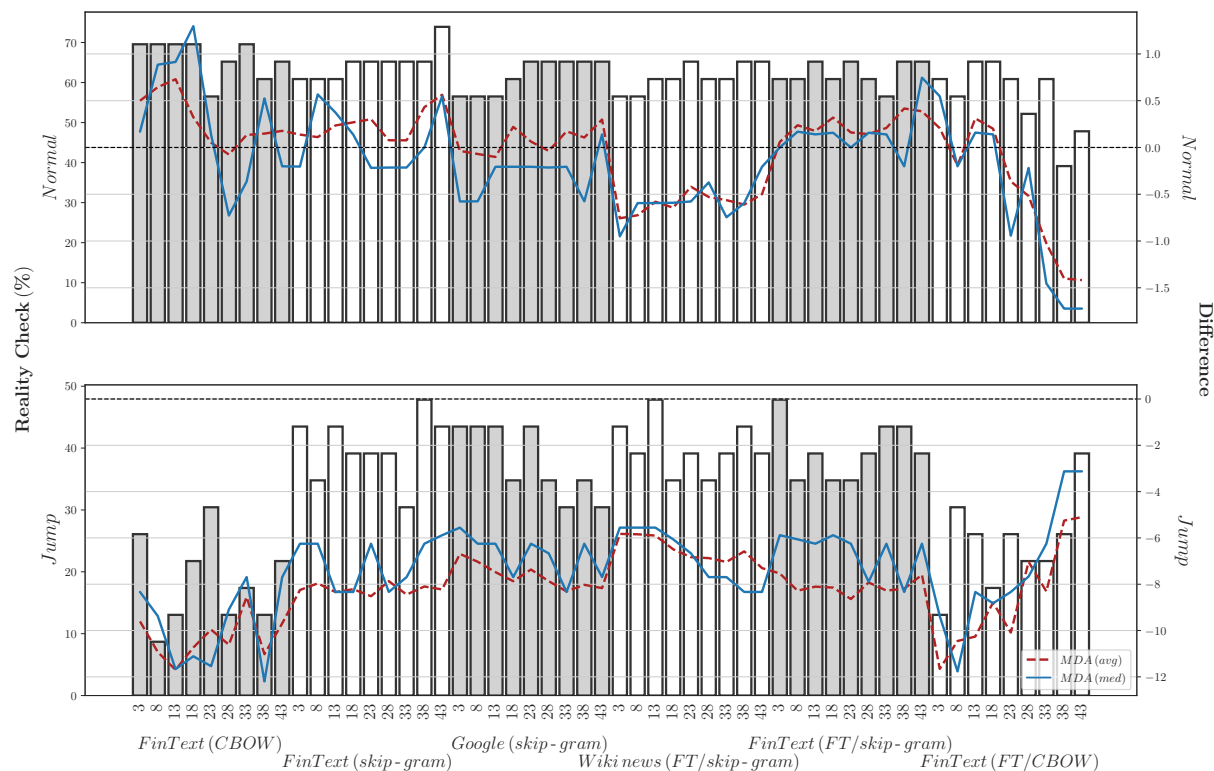


Figure A1: NLP-ML model performance - stock-related news (MDA)

Notes: The bar chart is the percentage of tickers with the outstanding performance considering the MDA loss function at the 0.05 significance level of the RC compared to the all HAR-family of models as the benchmark for every specified NLP-ML model. The values for the bar chart can be read from the left-hand axis. The dashed (solid) line represents the difference between the average (median) of the out-of-sample MDAs of the specified NLP-ML model with the CHAR model (the best performing HAR-family model in Rahimikia and Poon (2020a)) for 23 tickers (negative value shows degradation, and positive value shows improvement in performance). The values for the dashed and solid lines can be read from the right-hand axis. The horizontal dashed line represents no improvement.

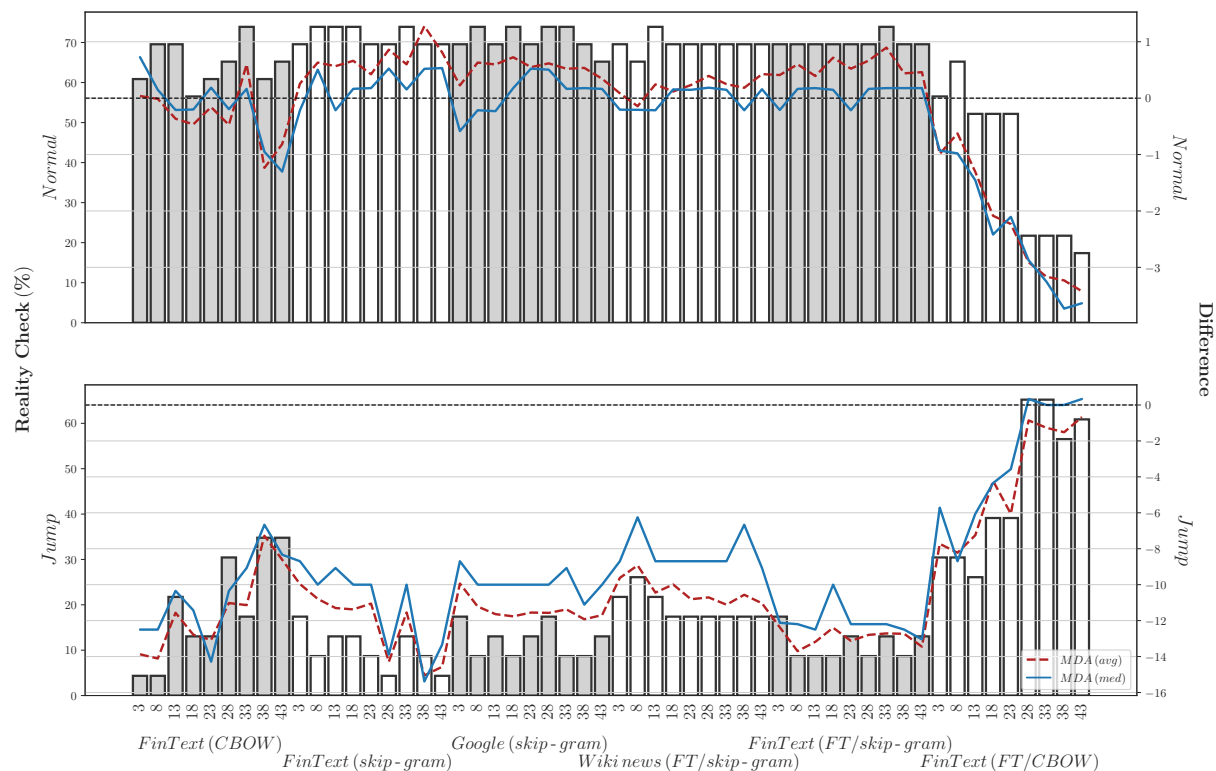


Figure A2: NLP-ML model performance - hot political news (MDA)

Notes: The bar chart is the percentage of tickers with the outstanding performance considering the MDA loss function at the 0.05 significance level of the RC compared to the all HAR-family of models as the benchmark for every specified NLP-ML model. The values for the bar chart can be read from the left-hand axis. The dashed (solid) line represents the difference between the average (median) of the out-of-sample MDAs of the specified NLP-ML model with the CHAR model (the best performing HAR-family model in Rahimikia and Poon (2020a)) for 23 tickers (negative value shows degradation, and positive value shows improvement in performance). The values for the dashed and solid lines can be read from the right-hand axis. The horizontal dashed line represents no improvement.

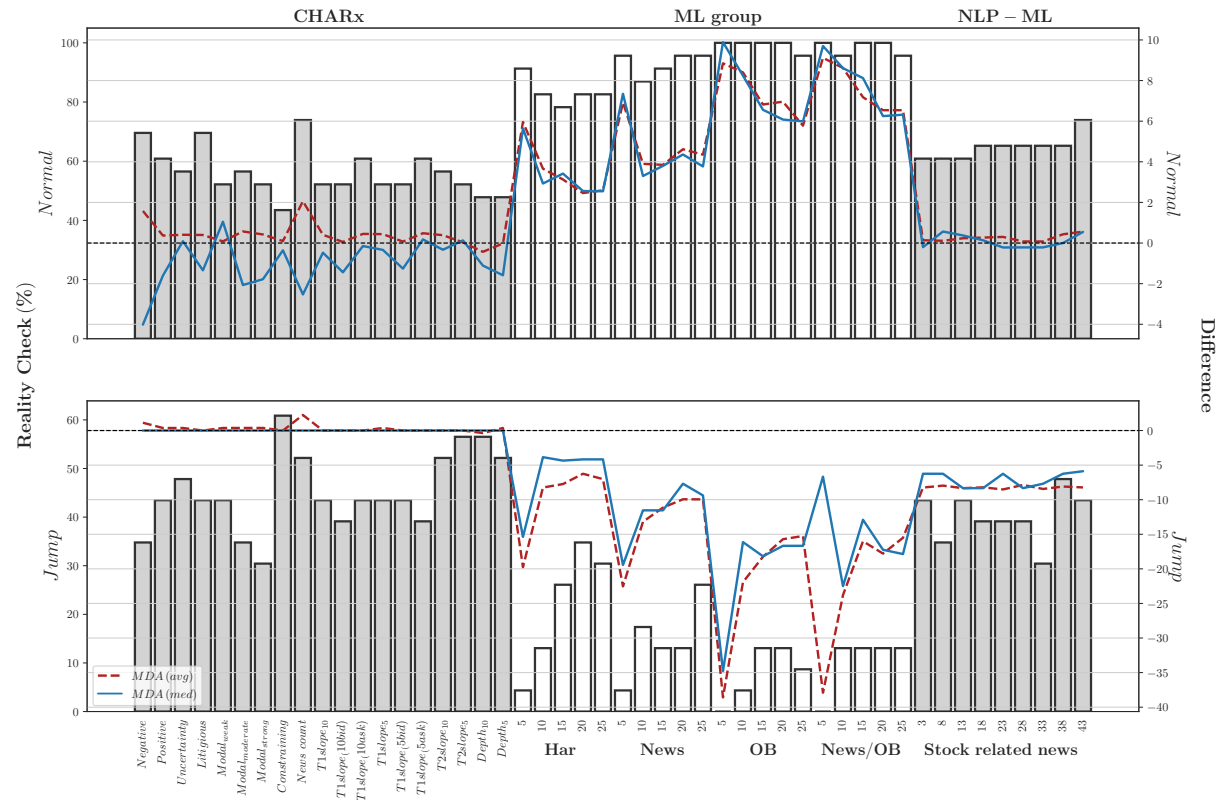


Figure A3: ML, extended CHARx, and NLP-ML models comparison (MDA)

Notes: The bar chart is the percentage of tickers with the outstanding performance considering the MDA loss function at the 0.05 significance level of the RC compared to the all HAR-family of models as the benchmark for every specified model. The values for the bar chart can be read from the left-hand axis. The dashed (solid) line represents the difference between the average (median) of the out-of-sample MDAs of the specified model with the CHAR model (the best performing HAR-family model in Rahimikia and Poon (2020a)) for 23 tickers (negative value shows degradation, and positive value shows improvement in performance). The values for the dashed and solid lines can be read from the right-hand axis. The horizontal dashed line represents no improvement.

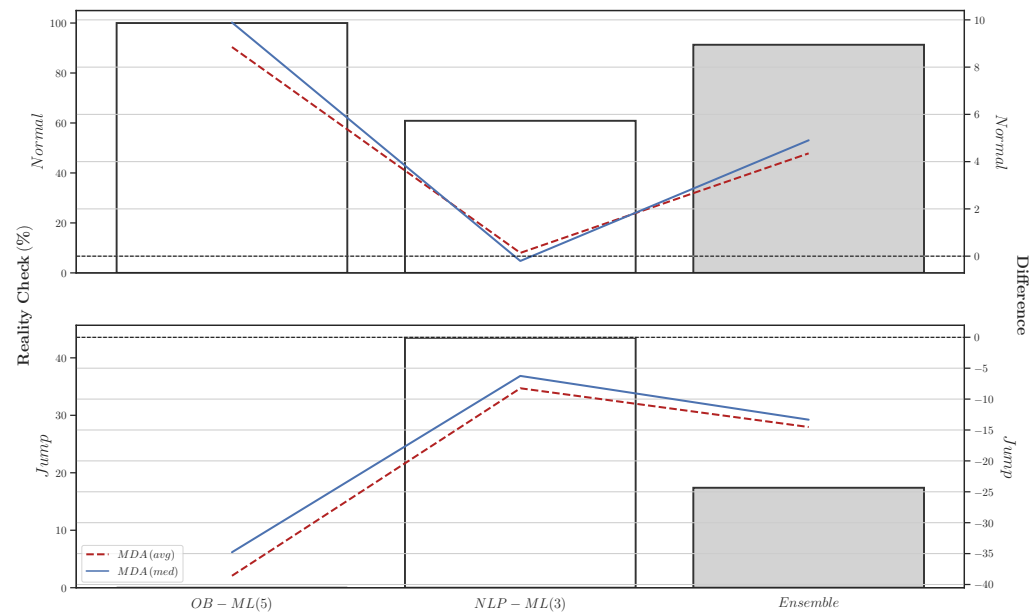


Figure A4: Performance of ensemble model, against OB-ML(5) and NLP-ML(3) - MDA

Notes: The bar chart is the percentage of tickers with the outstanding performance considering the MDA loss function at the 0.05 significance level of the RC compared to the all HAR-family of models as the benchmark for OB-ML model with five units as the chosen model from Rahimikia and Poon (2020b), NLP-ML model with three filters as one of the best performing models in this study, and the ensemble model. The values for the bar chart can be read from the left-hand axis. The dashed (solid) line represents the difference between the average (median) of the out-of-sample MDAs of the specified model with the CHAR model (the best performing HAR-family model in Rahimikia and Poon (2020a)) for 23 tickers (negative value shows degradation, and positive value shows improvement in performance). The values for the dashed and solid lines can be read from the right-hand axis. The horizontal dashed line represents no improvement.

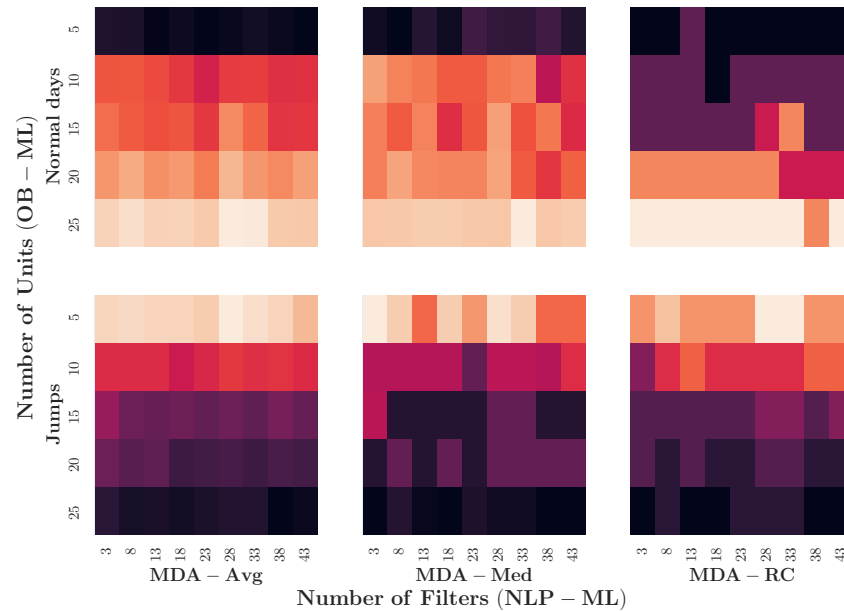


Figure A5: Ensemble models comparison - Number of Units vs. Filters (MDA)

Notes: The x-axis is the NLP-ML model with 3, 8, 13, 18, 23, 28, 33, 38, and 43 as the number of filters and y-axis is the OB-ML model in Rahimikia and Poon (2020b) with 5, 10, 15, 20, and 25 as the number of units. The first(second) row contains the results for normal volatility days(jumps). MDA-Avg(Med) is the difference between the average(median) of the out-of-sample MDAs of the specified NLP-ML model in the x-axis and OB-ML in the y-axis with the CHAR model (the best performing HAR-family model in Rahimikia and Poon (2020a)) for 23 tickers. MDA-RC is the percentage of tickers with the outstanding performance considering the MDA loss function at the 0.05 significance level of the RC compared to the all HAR-family of models as the benchmark for every specified NLP-ML model in the x-axis and OB-ML in the y-axis. Darker cell shows improvement, and lighter cell shows degradation in performance.

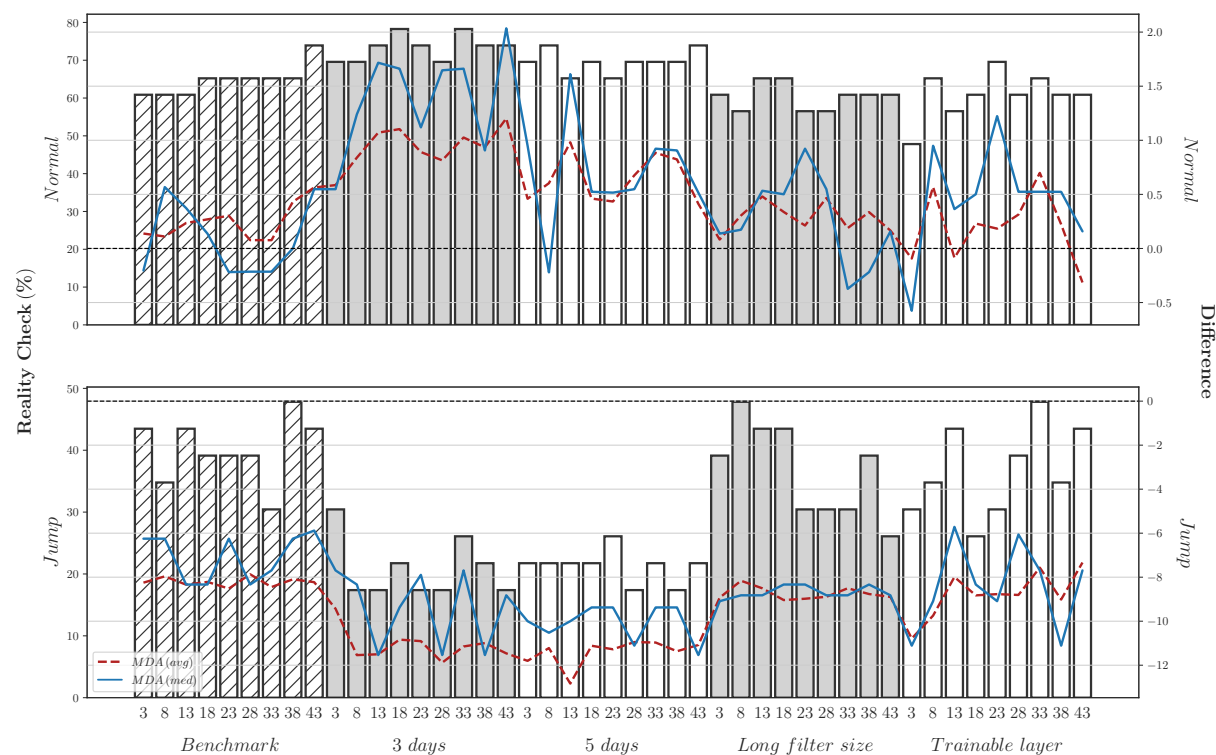


Figure A6: Robustness checks - MDA (performance comparisons)

Notes: The bar chart is the percentage of tickers with the outstanding performance considering the MDA loss function at the 0.05 significance level of the RC compared to the all HAR-family of models as the benchmarking. Benchmark group (hashed bars) is ‘FinText(skip-gram)’ group in Figure 5. For ‘3 days’ and ‘5 days’, the time duration of input data is changed from one day to three and five, respectively. For the ‘Long filter size’ group, 8, 9 and 10 are chosen as filter size (kernel size) instead of 3, 4, and 5. Furthermore, regarding the ‘Trainable layer’ group, the embedding layer of the NLP-ML model is set to trainable. The values for the bar chart can be read from the left-hand axis. The dashed (solid) line represents the difference between the average (median) of the out-of-sample MDAs of the specified model with the CHAR model (the best performing HAR-family model in Rahimikia and Poon (2020a)) for 23 tickers (negative value shows degradation, and positive value shows improvement). The values for the dashed and solid lines can be read from the right-hand axis. The horizontal dashed line represents no improvement.