Unsupervised Text Mining Techniques for Forecasting Crude Oil

by

Jacob Hazen

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master's of Arts

in

Economics specializing in Data Science

Carleton University

Ottawa, Ontario

ABSTRACT

Unsupervised Text Mining Techniques for Forecasting Crude Oil

A dissertation presented to the Faculty of the
of Public Affairs of Carleton University
Ottawa, Ontario

By Jacob Hazen

While it has been shown that news articles can influence the rationality of investors' decisions, the effect that news may have on commodity prices such as crude oil is uncertain. I explored Natural Language Processing (NLP) techniques to extract textual features from news articles and then constructed a "horse-race" among economic and tree-based machine learning methods to forecast weekly crude oil prices. I obtained two types of textual features, latent topics and sentiment probabilities, using two state-of-the-art NLP models: Latent Dirichlet Allocation (LDA) and a pre-trained version of Bidirectional Encoder Representations from Transformers (BERT) on a financial corpus. This paper introduced a novel forecasting strategy to calculate the out-of-sample (OoS) performance metrics of competing models. The evidence I found shows that textual features can improve forecasts of oil prices, however, textual features from news on their own are not sufficient for high forecasting accuracy.

Acknowledgements

First, I would like to thank Professor Ba Manh Chu for assisting with creation of this the thesis. Without your guidance and assistance on this thesis, it would have never been accomplished. I would also like to thank my best friend Montek Parmar for his careful edits to this thesis, as well as my sister Melissa Hazen. Most importantly, none of this would have happened without my family's support. Thank you to my girlfriend Melia Alcantara for her support, edits, comments on my thesis as well as words of encouragement throughout this journey.

Table of Contents

List of Figures

List of Tables

1 Introduction

Crude oil plays a critical role in the world's economy. Crude oil is one of the fundamental variables in generating macroeconomic projections and assessing macroeconomic risks (Alquist et al., 2013). In a study conducted by Hamilton (2009), the author provided evidence that the automobile industry's economic decline magnified the economic recession of 2008. This illustrates the influence crude oil has on the global economy, hence why macroeconomists, financial analysts, and policymakers race to forecast the price of oil. The literature pertaining to the forecast of crude oil prices is sizable. Forecasts such as that of Kilian and Murphy (2014) who fitted a structural model of the global market for crude oil that allows for shocks for flow demand and supply, and Mohan et al. (2019) who predicted multiple stocks including oil on the New York and Nigeria Stock Exchange using an Autoregressive integrated moving average (ARIMA) model. Forecasters have developed numerous traditional statistical and econometric models which can be clustered into three categories: structural, linear, and nonlinear time series models.

Oil prices are mostly affected by demand factors, supply factors, or political developments (Filis et al., 2011). While it has been shown that news and stock prices are closely correlated and that the news influences investors' stock market investments, the effect that news has on commodity prices such as crude oil is uncertain. The literature is limited on forecasting commodities using text mining techniques, despite the fact that it is recognized that oil prices are correlated with stock prices and stock prices are correlated with the news. This paper explores the question: can text from news articles forecast crude oil prices?

Obtaining information from unstructured text data is now achievable due to the constant advances in computational power and machine learning. Although, researchers are faced with one common problem when dealing with unstructured text

data: unlabelled data sets. Large labelled training sets are not easily obtainable for implementing NLP tasks for research and development due to the high cost and arduous task given the nature of annotating data. For this reason, in the NLP community, researchers are seeing a sudden shift toward pre-trained deep language modelling.

In this paper, I explore state-of-the-art unsupervised techniques for extracting features from unlabelled and unstructured news data to forecast the West Texas Intermediate (WTI) crude oil price. Specifically, I obtain sentiment and topic probabilities from the content of news articles relating to WTI crude oil. I show that NLP can be a useful intermediary step for economists, financial analysts, policymakers, and forecasters for empirical analysis. I also explore the predictive strength of these textual features with two tree-based algorithms and one statistical economic-based algorithm. These algorithms are E**x**treme **G**radient **B**oosting (XGBoost), Random Forest and Autoregressive moving average (ARMA) respectively. Tree-based algorithms are among the most popular algorithms used in machine learning given their versatility, accuracy, and stability. The foundation of the Random Forest algorithm and XGBoost is the decision tree. In simple terms, decision trees are non-parametric tree-like model that defines a set of rules created by training the tree on a data set to obtain a prediction.

Existing forecasting strategies may not take into account structural instability issues over time, so I employ a novel forecasting strategy (Chu and Qureshi, 2021) that compares OoS forecasting performance. This is an important step as it allows for fairness in assessing forecast performance across competing forecasts. For example, a particular model may be selected because of its top performance given a specific historical sample, however, it may underperform other models when given a sample on the recent data or a different historical sample.

The first key feature of my approach is that my method allows forecasters to extract textual features from the content of the news articles without having to make

theoretical assumptions on the data that require expert opinions. For example, Chen (2021); Ke et al. (2019); Luss and d'Aspremont (2015) formatted the sentiment task into a supervised task to train their NLP models by assuming all news changes the market in some way, either positively or negatively. This is a problem since not all news changes the price of the stock, and even more so, not all positive news means the stock will go up and vice versa. My solution is an unsupervised one, thus, it does not need to make these assumptions to extract sentiment from the text. My solution obtains the sentiment of the article as a probability that is either neutral, positive, or negative. The second key feature is the adoption of a pre-trained BERT on the financial corpus. This allows my sentiment analysis to use contextual embeddings trained on a financial setting instead of static embeddings or lexicon-based approaches which are tuned for common day language. The last key feature is that my time series makes use of latent topic features that are trained and tuned by the state-of-the-art LDA model.

1.1 Literature Review

The theory that news sentiments can predict market prices has been accepted for a long time. For instance, traders have made significant returns by basing their short-term investing strategy solely on news. Some of the earliest papers to implement news into forecasting markets explored the sentiment in the text regarding stock pricing dates. In the 1930s, Cowles (1933) manually read through The Wall Street Journal articles and classified if the article was either positive, negative, or neutral in sentiment. Cowles compiled 16 financial services and 20 fire insurance companies and found that the Wall Street Journal recommendations of stocks underperformed by 1.43% and 1.2%, respectively, compared to the average common stock. These results made it one of the most influential sentiment-based strategies at the time. However, the practicality of reading through thousands of papers is no longer sufficient.

Another influential paper by Tetlock (2007) implemented a pessimism-based strategy that measured the interaction between news media and the stock market. In addition, Tetlock found that lagged sentiment affected next-day returns. This strategy exhibited returns of 7.3% annually. However, Tetlock did not investigate the predictive power of positive sentiment within news articles.

When it comes to creating dictionary-based sentiment classifications, Loughran and McDonald (2011) found that 73.8% of words based on the Harvard-IV-4 word classification were misclassified as negative when applied to financial texts. Loughran and McDonald created a list of 2,337 words that had negative financial implications. They analyzed 10-K filings and found their proposed dictionary was better correlated with returns than the Harvard-IV-4. To summarize, Loughran and McDonald only ameliorated the problem of incorrect sentiment scoring of the existing lexicons but did not address the underlying problems of lexicon- and rule-based sentiment analysis.

The first problem of lexicon- and rule-based sentiment analysis is that it does not deal with out-of-vocabulary words, meaning it will not be able to assign a sentiment score to a word that is not already predefined. This leads to misclassification and bias of articles. Secondly, lexicon- and rule-based methods are not able to learn from the context of the sentence, semantic meaning, or the relationships between words, leading to ambiguity when assigning sentiment scores. Nevertheless, lexicon-based sentiment forecasting has shown promising returns. With the development of computational techniques in machine learning and Artificial Intelligence (AI), researchers have moved beyond lexicon-based sentiment models and towards pre-trained deep learning models that can produce contextualized word embeddings.

The two most common state-of-the-art algorithms for word embeddings that can capture the semantic similarity of words and different facets of the meaning of a word are Word2Vec (Goldberg and Levy, 2014; Mikolov et al., 2013b,a) and Global Vectors for Word Representation (GloVe) (Pennington et al., 2014). These algorithms

have achieved exceptional results in NLP tasks such as sentiment analysis, question answering, document clustering, paraphrase detection, and more. Word2Vec is an algorithm that uses a Neural Network (NN) to create word embeddings to learn semantic meanings between different words. GloVe is based on matrix factorization techniques on the word-context matrix. It first constructs a large matrix of words and context co-occurrence information. The drawback of both Word2Vec and GloVe is that they are context-independent and suffer from out-of-vocabulary words. They are context-independent because both algorithms produce a single vector for each word that combines different word meanings into a single vector. A popular example is "We went to the river bank" and "I need to go to the bank to make a deposit". The word "bank" has different meanings depending on the context. To represent the word "bank," more than a single word embedding vector would be required to represent the different meanings of words given the context. This is where Word2Vec and GloVe fail. They both try to represent the word embedding in a single vector for both contexts. This is the problem of polysemy—the coexistence of many possible meanings for a word based on context.

In recent developments, Embeddings from Language Models (ELMo) (Peters et al., 2018) was introduced to solve the problem of polysemy following an advancement that led to BERT (Devlin et al., 2019). ELMo and BERT incorporate context into word embeddings, replacing context-independent vectors with context-dependent vectors. This led to significant improvements on every NLP task. The main limitation of ELMo is a lack of ability to consider both left and right contexts of the target word at the same time. Even though uses a bidirectional Long Short Term Memory (LSTM) NN, it simply concatenates the left-to-right and right-to-left information, meaning that the word embedding cannot utilize both left and right contexts simultaneously. The advancement of BERT over ELMo is that it is a deep bidirectional model, capturing context from both left and right tokens simultaneously. In the current day, BERT is

the go-to language model for most NLP jobs because of its track-record on natural language tasks like General Language Understanding Evaluation (GLUE), Stanford Question Answering Dataset (SQuAD), and Situations With Adversarial Generations (SWAG), obtaining state-of-the-art results.

FinBERT (Araci, 2019) is a pre-trained version of BERT on a financial corpus that achieves state-of-the-art results for financial NLP tasks. To deal with specialized financial language, Araci found that BERT was not effective enough which led the author to pre-train BERT on the Reuters TRC2 and the Financial PhraseBank data set, achieving a 15% accuracy improvement in financial text classification tasks compared to the original BERT. Additionally, the FinBERT model achieved a 97% accuracy on the subset of the Financial PhraseBank with 100% annotator agreement for text classification. It has been widely used in the literature for applying contextualized embeddings to financial tasks (Chen, 2021; Lee et al., 2021; Mishev et al., 2020; Sonkiya et al., 2021). Sonkiya et al. (2021) finds that their Generative Adversarial Network, which uses the sentiment of news and headlines using FinBERT, outperformed traditional time series models like ARIMA. Chen (2021) introduced a new method that involves Recurrent Neural Networks (RNN) and FinBERT. Chen extracted FinBERT's contextual embeddings to use in his RNN. Chen found significant portfolio gains in trading simulations.

Topic modelling is used widely in the computer science field with constant advances to topic modelling research every day. The most used topic model to date is LDA (Blei et al., 2003). LDA is a generalization of the first probabilistic topic model called probabilistic latent semantic analysis (LSA) (Hofmann, 1999), which originated from the research of latent semantic indexing (LSI) (Deerwester et al., 1990). Topic modelling is predominantly used in unsupervised learning, however, researchers have modified topic models to work with supervised learning. Some of the well-known supervised topic models include supervised LDA, commonly known as sLDA (Mcauliffe

and Blei, 2007) or discriminative variation on LDA (DiscLDA) (Lacoste-Julien et al., 2008). Supervised topic modelling has been proposed as a way to guide the model through the problem that typical topic models face: latent topics found do not match the true topics. Of course, the usage of supervised topic modelling depends on the researcher's use-case; if the researcher has labelled data. In this paper, I explore unsupervised topic modelling—specifically LDA. Topic Modelling is used to single out latent topics in a piece of text and to derive hidden patterns exhibited by a text corpus.

Topic modelling's capabilities have brought research from a wide range of fields and topic modelling has even been used to forecast macroeconomic variables. For example, Larsen and Thorsrud (2019) decomposed news text using LDA into topics and then used these topics to choose articles to conduct sentiment analysis on. Another example is Ellingsen et al. (2020), who extracted 80 topics from a news corpus and then constructed a horse-race forecast among news and hard economic data. Bai et al. (2022), building on the analysis of Li et al. (2019), constructed topic intensity features from headlines of news text using Non-negative Matrix Factorization and sentiment intensity features using TextBlob. However, this approach was quite limiting in several ways. First, their decision to only obtain information from headlines limited the predictive potential of the model. Headlines can be biased and misleading or authors may emphasize minor aspects rather than a dominant point of the article to attract readers. Second, TextBlob is lexicon- and rule-based and it will ignore words that are not in its dictionary and consider only words and phrases that it can assign a polarity score. I identify these errors by using the pre-trained and fined tuned FinBERT for sentiment and LDA for extracting topics from the content of the article.

To my knowledge, this paper is the first to explore news text using FinBERT and topic modelling for commodity price prediction. The rest of the paper is organized as follows. In Section 2, I give a brief description of the models and methods explaining

how they work. In Section 3, I set up the models and present the methodology. Then in Section 4, I present the empirical results. In Section 5, I discuss the results, limitations, strengths, and future work of my paper. Finally, Section 6 concludes.

1.2 Objectives

The objective of this study was threefold:

1. To determine if the textual features extracted from news text add predictive performance to the statistical and tree-based forecasting methods.

2. To investigate how the different forecasting methods behave among different performance metrics and methods.

3. To discover the optimal forecasting method among competing models in the "horse-race".

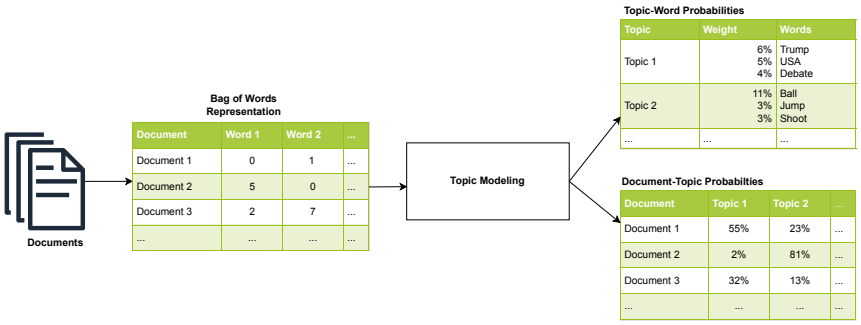
The forecasting results of my experiment suggest the following answers for the three objectives listed above. (1) The textual feature improved all the forecasting models compared to models without textual features except for the XGBoost algorithm. (2) All models performed fairly similar in terms of performance metrics, however, the tree methods performed the worse in terms of OoS R^2. (3) The best performing model was the rolling mean forecast, however among the forecasting models I implement in this paper, the Autoregressive moving average with e**X**ogenous input (ARMAX) model with textual features had the best overall performance.

2 Preliminary Models and Forecasting Methods

2.1 Topic Modelling

Throughout this paper, to keep terminology and language consistent, I consider an article to be a document, I will be using these terms interchangeably. A high-level overview of the process of topic modelling can be easily visualized in Figure 1. It takes in a collection of documents, then outputs the discovery of abstract "topics" within the documents. These topics are clusters of similar words, more specifically, a probability distribution over a fixed vocabulary. From the word cluster distribution, it is then possible to calculate the distribution of topics per document.

Figure 1: Topic Modelling Process

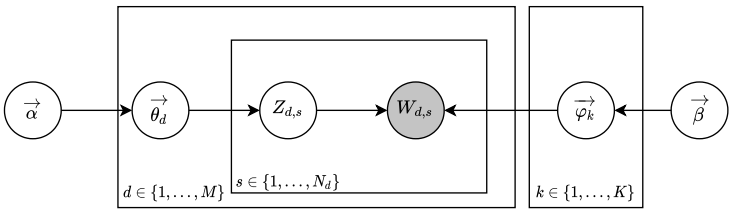


2.2 Latent Dirichlet Allocation

LDA is a generative mixed membership model for modelling sparse vectors of count data such as Bag-of-Words (BoW). It consists of corpus D consisting of M documents with word length of N_d (document d has word length N_d). The directed graphical model of LDA is given by Figure 2. Where each node is a random variable, an edge is a possible dependency, the shaded node is an observed variable, and the plates denote a replicated structure. What Figure 2 says is that each topic is characterized by a probability distribution over words, and each word in each document comes from a topic. As for the documents, they are defined as mixtures of latent topics and certain

words have a probability of occurring in specific topics more than others. LDA yields a vector containing the probabilities for each document belonging to a topic. LDA has three hyperparameters, α being the document-topic density factor, β being the topic-word density factor, and lastly K being the number of topics.

Figure 2: LDA Blueprint



LDA treats a document as a mixture of topics and the topic as a mixture of words. LDA follows this generative process:

1. Choose a $\theta_d \sim Dir(\alpha)$, where $d \in \{1,\ldots,M\}$ and $Dir(\alpha)$ is a Dirichlet distribution
2. Choose a $\varphi_k \sim Dir(\beta)$, where $k \in \{1,\ldots,K\}$
3. For each of the word positions d, s where $d \in \{1,\ldots,M\}$ and $s \in \{1,\ldots,N_d\}$
   (a) Choose a topic $z_{d,s} \sim$ Multinomial(θ_d)
   (b) Choose a word $w_{d,s} \sim$ Multinomial$(\varphi_{z_{d,s}})$

Where the total probability of the model is then given by the joint distribution:

$$P(\mathbf{W}, \mathbf{Z}, \theta, \varphi \mid \alpha, \beta) = \prod_{k=1}^{K} P(\varphi_k \mid \beta) \prod_{d=1}^{M} P(\theta_d \mid \alpha) \prod_{s=1}^{N} P(Z_{d,s} \mid \theta_d) P(W_{d,s} \mid \varphi z_{d,s}) \quad (1)$$

where $P(\varphi_k \mid \beta)$ is the corpus-topic distribution in the documents and $P(\theta_d \mid \alpha)$ is the document-topic proportions, both from a Dirichlet distribution. Next, take the θ_d proportions to generate the topics $z_{d,s}$ from a multinomial distribution $P(Z_{d,s} \mid \theta_d)$.

Using the percentages generated from φ_k and the multinomial distribution we obtain the words per topic $P(W_{d,s} \mid \varphi z_{d,s})$.

To use the LDA model given a collection of documents, it is possible to infer the hidden distributions: per-word topic assignments $Z_{d,s}$, per-document topic proportions θ_d, and per-corpus topic distributions φ_k. There has been many developments in unsupervised LDA inference and learning algorithms. These algorithms usually fall into one of three paradigms: maximum likelihood estimation, maximum a posteriori estimation, and Bayesian inference. Some of the most known algorithms to approximate the LDA posterior are mean-field variational methods (Blei and Jordan, 2004), expectation propagation (Minka and Lafferty, 2012), fast collapsed Gibbs sampling (Porteous et al., 2008), which is an improvement on the original collapsed Gibbs sampler (Griffiths and Steyvers, 2004), and collapsed variational inference (Teh et al., 2006). In this paper, I choose to use the collapsed variational Bayesian inference algorithm as it is the cheapest computationally (Špeh et al., 2013). The performance of these algorithms have been compared in Asuncion et al. (2012); Mukherjee and Blei (2009); Špeh et al. (2013).

2.3 Topic Model Evaluation

Determining the coherency of a topic model is a tough subject matter due to its unsupervised training process. There is no gold-standard way of evaluating topic models, most methods can be captured using one of the three paradigms: eye-balling models, extrinsic evaluation and intrinsic evaluation. The most common approach to assess the quality of topics generated by the topic model is the eye-balling model. The eye-balling model is commonly performed by inspecting the most probable words from each topic. A major problem with this assessment is the repetition of the laborious work of manually assessing each topic and that the evaluation will be biased and subjective to the researcher's interpretation of said topics. Extrinsic evaluation

is typically done by measuring the performance on the task at hand, for instance, information retrieval or document classification. This evaluation method often needs a labelled data set which researchers may not have. For these reasons, this paper will focus on intrinsic methods for evaluating topic models.

A common way to evaluate any sort of probabilistic model is to measure the log-likelihood of a held-out test set, in other words, perplexity (Wallach et al., 2009). In simple language, it is a measurement of how well the probability model predicts a sample. Perplexity defined in Equation 2:

$$perp(q, x) \coloneqq 2^{\frac{1}{|x|} \sum_{i=1}^{|x|} \log_2 p(x)} \tag{2}$$

Where $|x|$ is the cardinality of the set of held-out documents, and q is the model that is tested. However, many researchers have identified problems with using perplexity as a metric to evaluate the models' performance on identifying latent topics. The problems are mostly situated on the fact that perplexity is a predictive evaluation metric—assessing solely on the ability to predict the held-out set; perplexity does not capture semantic meaning between words. Chang et al. (2009) showed that oftentimes an increase in perplexity score is negatively correlated with their discretion of human interpretable topics; models that achieved a better perplexity score had less interpretable topics. This finding suggests perplexity may not be a great measure for topic modelling. Nevertheless, quantitative evaluation for topics modelling has been a long-standing discussion in the NLP community, at present, perplexity is still a widely used quantitative measure.

In later developments in intrinsic metrics, topic coherence (Newman et al., 2010) improves on the limitations that perplexity possesses by attempting to capture the semantic nature of the learned topics. Quantifying coherency in language has been a well-established problem that has attracted researchers from diverse disciplines. Some measures are Point-wise Mutal Information (PMI) and Normalized Point-wise

Mutual Information (NPMI) (Bouma, 2009).

$$PMI(w_i, w_j) = \log \frac{P(w_i, w_j) + \epsilon}{P(w_i) \cdot P(w_j)} \tag{3}$$

$$NPMI(w_i, w_j) = \frac{PMI(w_i, w_j)}{-\log(P(w_i, w_j) + \epsilon)} \tag{4}$$

Probabilities are estimated based on word co-occurrence counts.

2.4 Bidirectional Encoder Representation from Transformers

BERT is a Transformer language model that consists of a set of Transformer encoders stacked on top of each other. A Transformer is an attention-based deep learning model that is used for modelling sequential information. BERT training is performed in two stages: pre-training and fine-tuning. BERT is pre-trained on two unsupervised tasks: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). The fine-tuning stage is for the researcher's particular use-case.

MLM can be thought of as a fill-in-the-blank task meaning BERT is given a sequential input up to 512 tokens long, then it replaces 15% of the words with a [MASK] token. BERT optimizes the weights inside its architecture to predict the original words of the masked tokens based on the context given by the other words in the sequence.[1] NSP receives a pair of sentences as inputs and is then trained to predict if the second sentence is probable given the first sentence. This is important for capturing the understanding between sentences. BERT simultaneously trains MLM and NLP together.

In the fine-tuning stage, it is common to add a NN layer on top of the model (after the Transformer encoders). For most fine-tuning tasks, the layer needs a specific input format, for instance, a singular embedding vector. This is usually extracted from the

[1]BERT implementation of MLM is slightly more elaborate due to mismatching between fine-tuning and MLM and does not replace all of the 15% masked words. Refer to Devlin et al. (2019)

classification [CLS] token (see Figure 3), which is added to the beginning of the input sequence. This [CLS] token is the word-sequence representation—embedding token that represents the sequence of words. For all tasks that involve classification, including next sentence prediction, we use the [CLS] as input to our fine-tuning task.

Figure 3: BERT Architecture

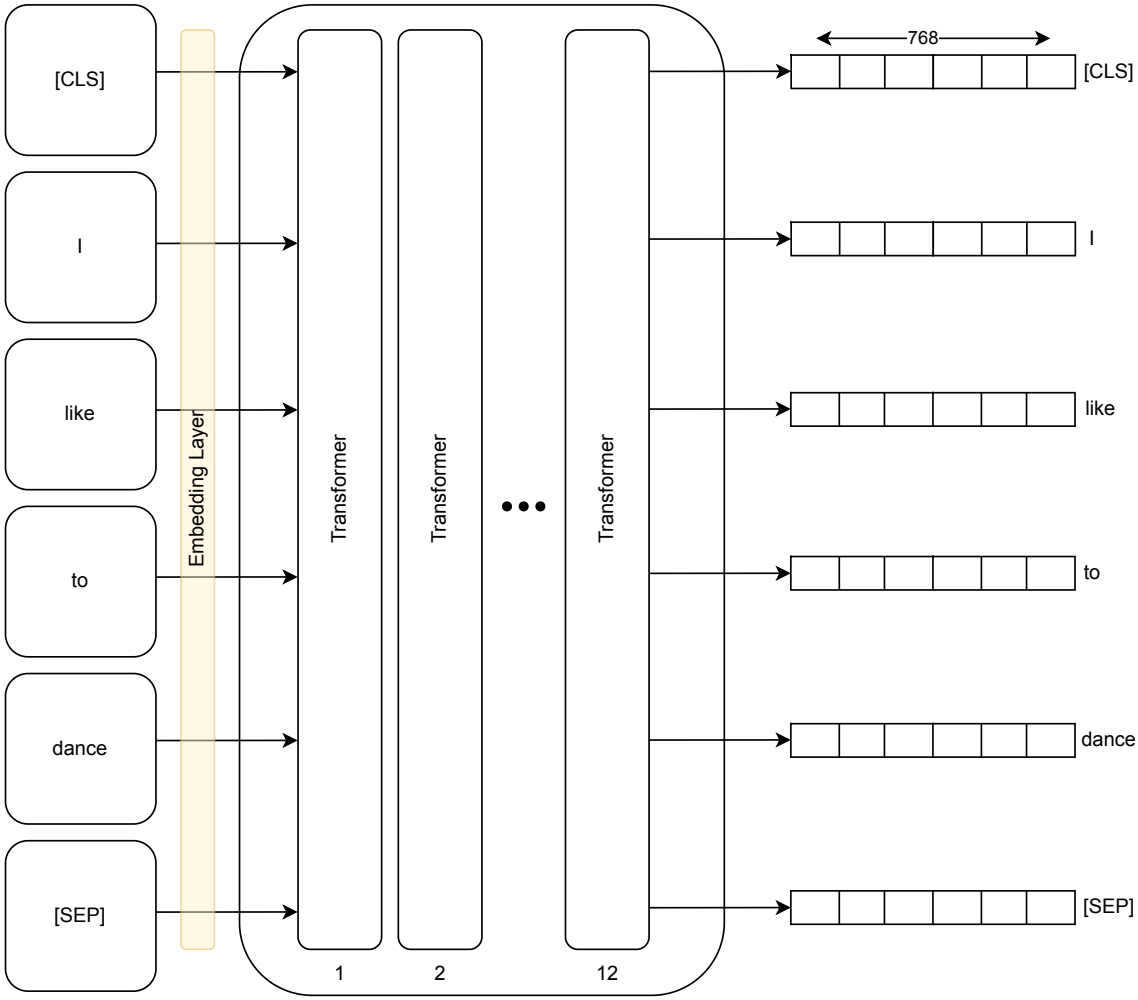


Figure 3, provides a high-level overview of BERTs architecture. The input starts from the left as tokens and makes its way through the embedding layer that matches the token to its respective WordPiece word embedding vector. These pre-embedded vectors pass through BERTs 12 layer transformer blocks then outputs the contextualized word embedding vectors, each with a vector length of 768.

2.5 Random Forest

The Random Forest algorithm proposed by Breiman (2001) is an ensemble learning method for classification and regression that trains multiple decision trees (in my case, regression trees) grown independently using sub-samples drawn from a sample. It takes a majority vote for classification tasks and the average for regression tasks. Decision trees are prone to over-complex trees that directly affect the generalization of the data, thus having high variance. However, the high variance that decision trees are prone to can be significantly reduced by random forests ensemble learning method *bagging* (**B**ootstrap **Agg**regat**ing**). Another way to reduce the variance is the proper tuning of the hyperparameters (see B.1)

To build a simple random forest:

1. Draw a bootstrap sample with replacement that is less than or equal to the sample size

2. Start building the Decision Tree:

   (a) Without replacement, randomly select f features where f is a subset of the total features F ($f \subseteq F$).

   (b) Split at the node using f features selected.

   This means

3. Repeat steps a and b for S independent training sets, creating $T^1(X), \ldots, T^S(X)$ trees

4. Average the trees, $T_S(X) = \frac{1}{S}\sum_{s=1}^{S} T^s(X)$

An important property to note in the last step, by the Central Limit Theorem, averaging the trees together, $T_S(X) = \frac{1}{S}\sum_{s=1}^{S} T^s(X)$, will drive the variance down to

0. ($\text{Var}(T_S(X)) = \frac{1}{S}\text{Var}(T^s(X)) = \frac{\sigma^2}{S}$). This is the main motivator behind Random Forests—it reduces the variance in the model.

2.6 Gradient Boosting

Like the Random Forest algorithm, the base of the Gradient Boosting algorithm is an ensemble of decision trees. However, unlike the Random Forest algorithm, it uses the ensemble method called *boosting*. Boosting primary goal works to reduce bias and variance by using several weak learner models to create a stronger model. Suppose we have the input data $\{(x_i, y_i)\}_{i=1}^n$ and a differentiable loss criteria $L(y, f(x))$ (for regression trees it is common for the loss function to be the sum of squared residuals multiplied by one half).

How the Gradient Boosting algorithm works which largely follows from Hastie et al. (2009):

1. Initialize the model with a constant value: $f_0(x) = \arg\min_\gamma \sum_{i=1}^n L(y_i, \gamma)$
2. Build trees $m = 1, 2, \ldots, M$:
   (a) For observation $i = 1, 2, \ldots, N$ compute the Pseudo Residuals

$$r_{i,m} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)} \tag{5}$$

   (b) Fit a regression tree targeting the Pseudo Residuals values, $r_{i,m}$, and create tree leafs $R_{j,m}$, for $j = 1, 2, \ldots, J_m$
   (c) For $j = 1, 2, \cdots J_m$ compute:

$$\gamma_{i,m} = \arg\min_\gamma \sum_{x_i \in R_{i,m}} L(y_i, f_{m-1}(x_i) + \gamma) \tag{6}$$

(d) Update with shrinkage (Often referred to as the *learning rate*, η):

$$f_m(x) = f_{m-1}(x) + \eta \sum_{j=1}^{J_m} \gamma_{i,m} I(x \in R_{j,m}) \tag{7}$$

3. Output $\hat{f}(x) = f_M(x)$

In this paper, I use Chen and Guestrin (2016) popular version of the Gradient Boosting algorithm, XGBoost. XGBoost is a regularizing gradient boosting algorithm, it follows the same principles of gradient boosting, however differs slightly to control overfitting and speed up computational performance. To support overfitting of the algorithm described above, the XGBoost algorithm defines a new objective function which consists of two parts: the loss function and the regularization term:

$$\text{obj}(\theta) = L(\theta) + \Omega(\theta) \tag{8}$$

The regularization term is added to control the complexity of the model—which helps with overfitting[2][3]. The XGBoost algorithm is engineered to optimize the speed in the Gradient Boosting algorithm. The algorithm defined above is sequential since Equation 7 requires the previous tree to be built. Therefore it is not possible to parallelize the algorithm, however, it is possible to parallelize the building of decision trees—XGBoost takes advantage of this. Similar to the Random Forest, proper hyperparameter tuning of the XGBoost algorithm helps lower overfitting. However, the regularization that the algorithm provides means that it has many more hyperparameters to tune. (see B.2)

[2]Overfitting is a situation where the in-sample performance is better than the OoS performance
[3]For details and a list of differences, visit the official documentation page

3 Methodology

First, in this section, I show general common practice pre-processing steps that are required to reduce noise in NLP models. Second, I step through the process of training LDA models. Additionally, I show how to perform hyperparameter tuning in topic modelling to obtain the most coherent topics. Third, I show a method of dealing with long text that exceeds BERT's 512 token limit within BERT's architecture. Finally, to forecast oil prices, I implement a recursive OoS forecasting strategy to access fairness among competing forecasting models.

3.1 LDA Pre-processing

It is common knowledge that BERT can handle punctuation (Ek et al., 2020), therefore, it is in the best interest of the language model to avoid the usual pre-processing steps that remove punctuations, stop words, lemmatization, rare words, and common words because BERT learns from the context when generating word embeddings. However, topic models need a more rigorous pre-processing step than BERT. Topic models, such as LDA, do not use context when generating topics. Instead, topic models use word co-occurrences, therefore normalization of the corpora can reduce the probability space of the model—by reducing the vocabulary size.

I loosely follow Schofield et al. (2017) recommendations for processing text for a LDA model:

1. Removal of stop words has little to no impact on the inference of topics on non-stop words. Except for the case of extremely frequent stop words.

2. Morphological methods such as lemmatization and stemming could potentially harm the LDA model.

The first step is to split the articles into word tokens, combining each token into

a list[4]. In addition, I expand contractions such as, "it's" to "it is", "didn't" to "did not", I do this to reduce the size and redundancy of my vocabulary of words, and make computation slightly cheaper. Next, I delete numbers, punctuation, stop words, special symbols, and non-English words as they tend to have lesser importance for the goal of topic modelling. To note, Schofield et al. (2017) reports that stop words have little to no effect on the inference, however, I remove stop words when previewing contents of the topics. The second step in most NLP pre-processing tasks is to lemmatize or stem the word tokens into their base word. Schofield et al. (2017) recommends not using stemming or lemming methods since topic model inference places words that share the same morphological roots in the same topics. They argue it is redundant and could potentially damage the topic model. However, I do compute and compare the topic coherence of models with this processing step.

Next, I compute n-grams for the corpus and add them to my dictionary.[5] An n-gram is a sequence of n adjacent words that commonly occur together. This is important because I want the model to treat some n-gram words as a one-word token. For example, "United States" will be tokenized as ["United", "States"], which does not make much sense as it will be treated as two separate identities. I want the model to treat the word as a whole, thus processing the phrase together as ["United_States"]. I apply a concatenation of the n words with "_" as the separator. However, it is very memory intensive to compute all n-grams for all the words in the dictionary. A method to compute n-grams efficiently is to first find words that appear frequently together and infrequently in other contexts. "Standard and Poor" or "West Texas Intermediate" appear less frequently in other articles compared to "of the" but still frequently together. I follow Mikolov et al. (2013a) simple data-driven approach for

[4]I use the natural language toolkit library to tokenize, lemmatize Bird et al. (2009)
[5]I use the gensim library to compute n-grams (Rehurek and Sojka, 2011)

computing phrases based on unigram and bigram counts.[6]

$$score(w_i, w_j) = \frac{count(w_i w_j) - \delta}{count(w_i) \cdot count(w_j)} \cdot sizeofvocabulary \tag{9}$$

Where δ is to prevent too many phrases consisting of very infrequent words to be formed. It can also be seen as the minimum total word count of $count(w_i w_j)$ for the two adjacent words to be considered a bigram. Mikolov labels δ as the discounting coefficient. I set $\delta = 20$ and choose the threshold score to be 10. Bigrams that are chosen above a certain threshold are then considered phrases and then added to my word tokenized documents.

To prune the corpus, I first compute the document frequency DF_w for each word. By computing the cardinality of the set of documents that contain word w, I obtain n_w as document weight given by Equation 10 where d is a single article in the total number of articles M.

$$n_w = |\{d \mid d \in M \wedge w \in d\}| \tag{10}$$

$$DF_w = \frac{n_w}{M} \tag{11}$$

Next, calculate the set of words S for each document which is the result of filtering out words that occur in less than 30 documents and more than 50% of the documents to remove rare and common words.

$$S = \{w \mid DF_w < \frac{1}{2}\} \cap \{w \mid n_w \geq 30\} \tag{12}$$

[6] I use Gensims Phraser method

3.2 LDA Training and Hyperparameters

After data pre-processing it is important to convert the articles into a Bag-of-Words (BoW) format since LDA requires the data to be in count data form[7]. BoW is a way to represent text in a document by counting the occurrence of words within a document by creating a vector of word counts. After BoW is performed, it is important to carefully select the three hyperparameters in the LDA model: the document-topic density factor α, topic-word density factor β, and the number of topics K. Increasing α is expected to decrease the sparsity of documents to topics. It can be thought of decreasing the penalty for documents having many topics. A decrease in β decreases the words per topic, in other words, increasing the penalty for topics having many words.

In choosing the right α and β parameters, I perform two methods of hyperparameter tuning. First, I follow Griffiths and Steyvers (2004) strategy to fix α and β, then explore the consequences of varying the number of topics K. Griffiths and Steyvers had a corpus size of $28,000$ documents and a vocabulary size of $20,000$. They evaluated topics using the hyperparameters: $K = [50, 100, 200, 300, 400, 500, 600, 1000]$, $\beta = 0.1$, and $\alpha = 50/K$. In comparison, my corpus is approximately 43% the size of Griffiths and Steyvers, I also expect my distribution of topics in each document to be sparse, therefore I set $\beta = 0.1$ and $\alpha = 5/K$. Next, I construct a search over K starting at 20 and ending at 40, increasing K by 1 at every step. At each step, I evaluate the four-stage topic coherence pipeline implemented by Röder et al. (2015). The second method is allowing the automatic tuning of the α and β which is automatically done by Gensim then vary K topics.

To evaluate the topic model results I compute the topic coherence framework which captures semantic similarity between high scoring words in a topic. This framework is often referred to as the *coherence pipeline* (Röder et al., 2015). Equation 13 is the

[7] This is done using Gensim

coherence pipeline I use in this paper:

$$C_{NPMI} = (S, NPMI, \sigma) \tag{13}$$

where

$$\begin{aligned} S &= \{(W', W^*) \mid W' = \{w_i\}; W^* = \{w_j\}; w_i, w_j \in W; i \neq j\} \\ PMI &= \log \tfrac{P(W', W^*)+\epsilon}{P(W') \cdot P(W^*)} \\ NPMI &= \tfrac{PMI}{-\log(P(W', W^*)+\epsilon)} \\ \sigma &= \text{arithmetic mean} \end{aligned}$$

In the first stage, I segment the words into word pairs. Second, I calculate a confirmation measure, $NPMI$, which uses word probabilities, I also compute word probabilities on the segmented words on a held-out test corpus. From Röder et al., I construct the best performing coherence measure C_{NPMI} when faced with a higher sliding window.[8] C_{NPMI} is the topic coherence pipeline based on the normalized point-wise mutual information measure (Bouma, 2009). Lastly, I take the mean denoted by σ.

3.3 FinBert Pre-processing

FinBERT is based on the BERT architecture, therefore requiring the same processing steps as BERT. BERT's framework suffers from an extreme limitation, its architecture only allows for 512 tokens to be processed at a time, meaning that long texts don't work under BERT's framework. This is because of BERT's adoption of the self-attention mechanism—scales computation and memory quadratically. Recent approaches to this problem fall under two methods: constructing the long Transformer (Beltagy et al., 2020) or a concatenation method of BERT (Pappagari et al., 2019). However, these methods are out of the scope of this research, thus I will be

[8]The C_V coherence measure had the best performance, however, there are known issues with this coherence measure discussed at this link, as a result Röder et al. does not recommend using it. So I default to the next best coherence measure. C_P, shown to be the next best result when faced with a lower sliding window. When faced with a higher sliding window size, C_{NPMI}, shown better results

using a dynamic aggregation method by chunking the text into tokens that is small enough to feed into FinBERT, then take the mean of the sentiment probabilities (see Appendix B.3 for technical details of the method).

3.4 Forecasting Strategy

To evaluate the OoS performance of a forecasting model over time, I construct a recursive rolling window sub-sample estimate with cross-validation. This method is inspired by Chu and Qureshi (2021)[9]; a high-level overview of the strategy can be viewed in Figure 4. This data-driven approach allows the hyperparameters to dynamically adjust at every forecast to optimize forecast predictions.

How this strategy operates is that it splits the sample size into fixed-length sub-samples, $i = 1, \ldots, N$. (I chose 285 sub-samples with 365 observations per sub-sample). At each sub-sample i, the strategy splits the data again into fixed-length *rolling windows*. (I use 5 rolling windows). The rolling window slides across the time sequence creating multiple sub-samples (or "cutouts") within the sub-sample. As the rolling-window slides across the time sequence, it breaks the cutouts into train, validation and testing sets (I use a 90% training split). Next, I perform a *walk-forward validation* using a grid search. A walk-forward validation is used to determine the optimal parameters in an OoS validation set. The model is trained with the in-sample data and then tested on a portion of the validation set, where the results are recorded. The in-sample training set is then shifted forward by the number of periods used on the validation set, and the process is repeated. The model that had the best result is then used for the OoS testing set. The performance metric used to determine the best hyperparameters for forecasting is the OoS Root Mean Squared Error (RMSE). I also evaluate two other performance metrics: OoS Mean Absolute Error (MAE), and OoS R^2. Calculations of these metrics follow from Chu and Qureshi (2021).

[9]I thank Dr. Ba Chu, my supervisor, for suggesting this forecasting strategy to evaluate OoS predictions

Let $\hat{Y}_{i,t+h}, t = \mathcal{T}_{1,i}, \ldots, \mathcal{T}_i - h$ represent h-period ahead predicted forecast in sub-sample i, where the actual observations are $Y_{i,t+h}, t = \mathcal{T}_{1,i}, \ldots, \mathcal{T}_i - h$. The $\mathcal{T}_i$ represents the fixed size of sub-sample i. Thus, I have $\mathcal{T}_i - h - \mathcal{T}_{1,i} + 1$ rolling windows. Therefore I can write the OoS performance metrics as:

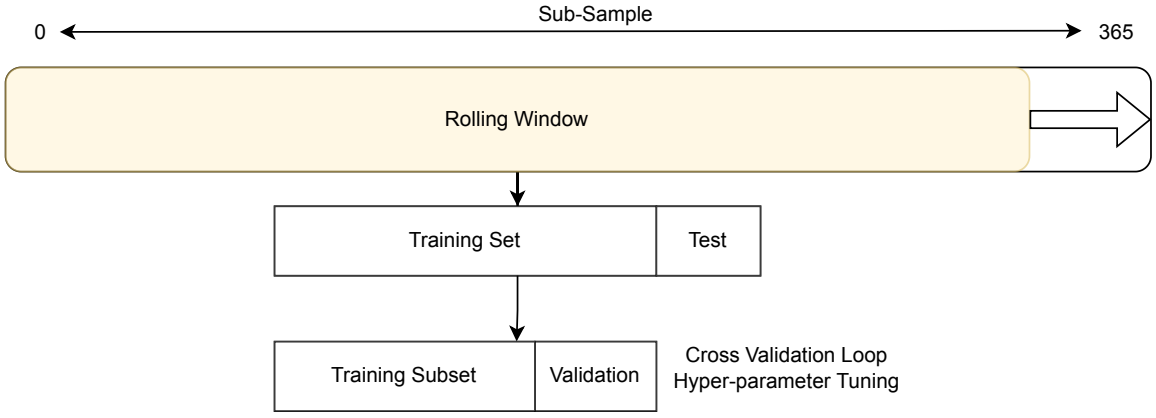
$$RMSE_i = \sqrt{\frac{1}{\mathcal{T}_i - h - \mathcal{T}_{1,i} + 1} \sum_{t=\mathcal{T}_{1,i}}^{\mathcal{T}_i - h} (Y_{i,t+h} - \hat{Y}_{i,t+h})^2} \tag{14}$$

$$MAE_i = \frac{1}{\mathcal{T}_i - h - \mathcal{T}_{1,i} + 1} \sum_{t=\mathcal{T}_{1,i}}^{\mathcal{T}_i - h} \mid Y_{i,t+h} - \hat{Y}_{i,t+h} \mid \tag{15}$$

$$R_i^2 = 1 - \frac{\sum_{t=\mathcal{T}_{1,i}}^{\mathcal{T}_i - h} (Y_{i,t+h} - \hat{Y}_{i,t+h})^2}{\sum_{t=\mathcal{T}_{1,i}}^{\mathcal{T}_i - h} (Y_{i,t+h} - \bar{Y}_{i,t+h})^2} \tag{16}$$

An important property to note about the OoS R^2 is that $\bar{Y}_{i,t+h}$ is the rolling mean forecast. If the value of $R^2 > 0$ then the forecast is better than the rolling mean forecast (in terms of the RMSE). If the value of $R^2 < 0$ then the rolling mean forecast is better than the forecast; the OoS R^2 value can range from $[-\infty, 1]$.

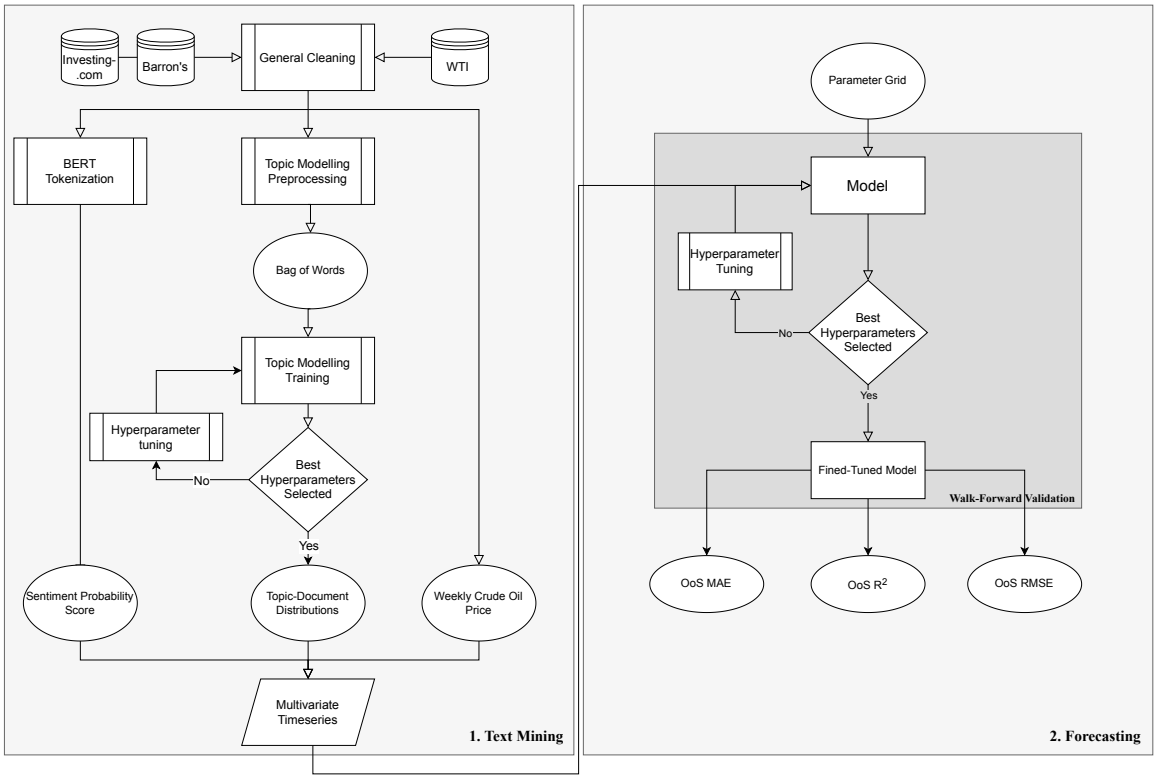
Figure 4: Forecasting Strategy



4 Empirical Analysis

In this section, I apply my text mining framework to explore the predictive strength of textual features to crude oil prices. More specifically I explore the sentiment and latent topics in an article through state-of-the-art NLP methods. A high-level overview of the framework is given by Figure 5. In the left rectangle, I apply my text mining framework and use the NLP models to extract textual features to use for my time-series forecast. In the right rectangle, I apply the OoS recursive rolling window sub-sample estimate with cross validation then output the three OoS performance metrics: MAE, R^2, and RMSE.

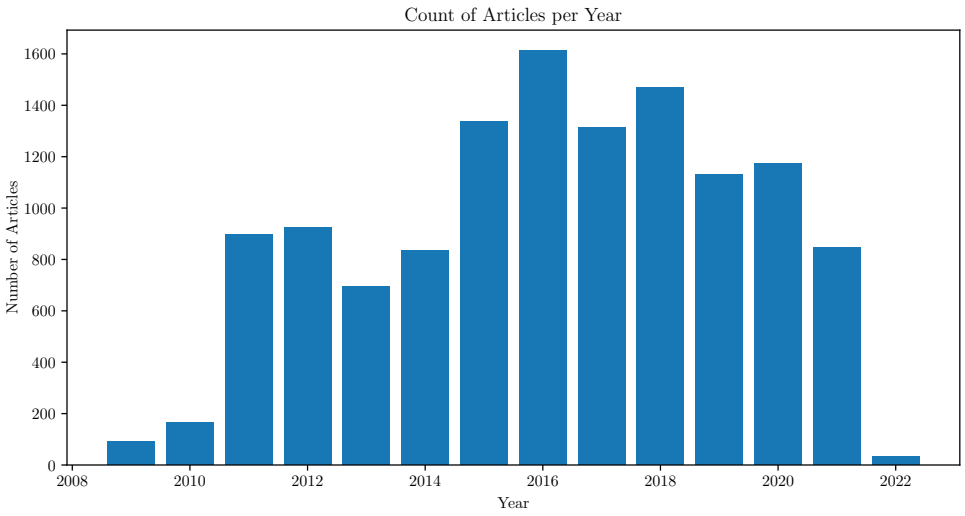
Figure 5: Crude Oil Forecasting Framework



4.1 Data and Pre-processing

The text data set is collected from Barrons and Investing.com. Barrons is an American newspaper company published by Dow Jones & Company. Barron's is a reliable and leading source of financial news, providing in-depth analysis and commentary on stocks, investments and how markets are moving. Investing.com is a reputable financial platform and news website; one of the top three global financial websites in the world. For credibility reasons, I chose these newspapers companies as data sources. For web-scraping, I use an open-source web-crawling framework: Scrapy. I combine Scrapy with the Python version of Selenium to render and interact with the javascript in the website.[10] I obtain a data set from Barron's with 10614 unique containing news feeds from September 9^{th}, 1997 to November 11^{th}, 2021. From Investing.com I obtain a data set from July 24^{th}, 2009 to January 18^{th}, 2022 containing 32,087 articles.

Figure 6: Count of Oil Articles Per Year

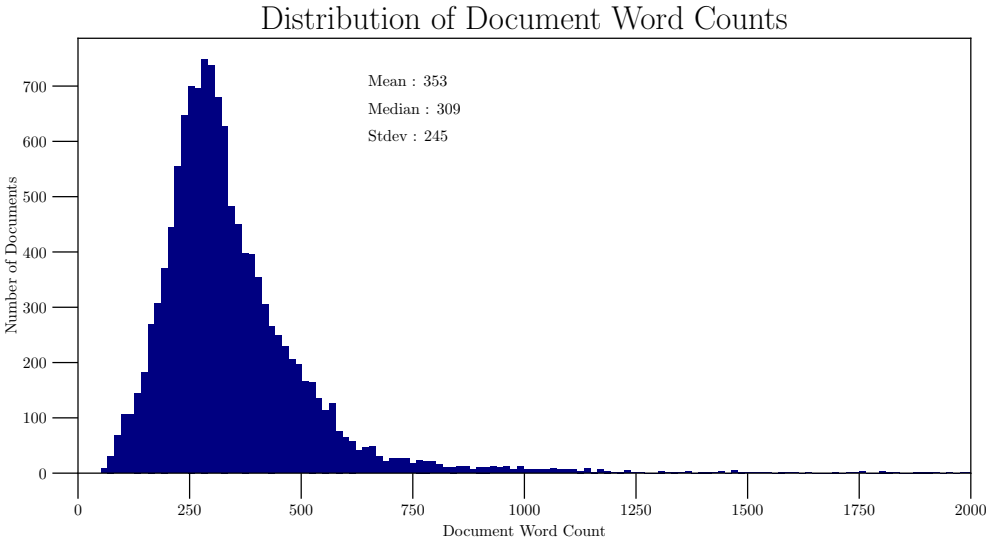


Immediately after the collection of the news articles, I perform data cleaning by removing junk or noise in the text. This was performed by using regular expres-

[10]More information about Scrapy can be found at this link and information about Selenium can be found at link.

sions. Some examples include external website links, emails, additional and random punctuation, and embedded sub-articles that might have been collected by the web scraper. I also compute the word count size of the articles, a visual check of the outliers determined that there was only one major outlier collected for each news outlet. Barron's had an outlier with over 175,000 words in the document, and 25,000 words for Investing.com; from manual inspection of the outliers, they were deemed unimportant and removed. I also remove useless articles, these articles were either only videos or reporting closing prices for the day. Abbreviations such as "S&P", "U.S." were transformed into their original forms by using a predefined dictionary to reduce the size of the vocabulary and noise.

Figure 7: Words per Document Distribution



Upon data exploring of the articles, the majority were unrelated to crude oil only briefly mentioning crude oil. For example, the article would be talking about cryptocurrency but mention the price of the oil in one or two sentences. To combat this problem, I count the number of times "oil" is mentioned in an article, specifically 61% of articles contained the keyword "oil" less than 5 times. I found from inspection of the data that if "oil" was mentioned 5 or more times, it removed the problem of articles

Table 1: Data Cleaning

Filter	Barron's	Removed	Investing.com	Removed	Total
Initial sample size collected from webscraper	10,605		32,087		42,691
Outlier	10,604	1	32,086	1	42,689
Removal of useless articles	10,126	478	32,086		42,231
Removal of articles containing "oil" less than 5 times	3,215	6,911	10,165	21,922	13,380
Removal of Barron's articles before 2009-07-24	2,363	852	10,165		12,528

that only briefly mentioned oil. The final sample contained 12,528 articles. Features that were extracted from the articles contained: publish date, headline, author, and text content of the article. The date and content are necessary to collect, the others were used for data validation purposes. Figure 7 shows visually the distribution of article word counts after performing the cleaning.

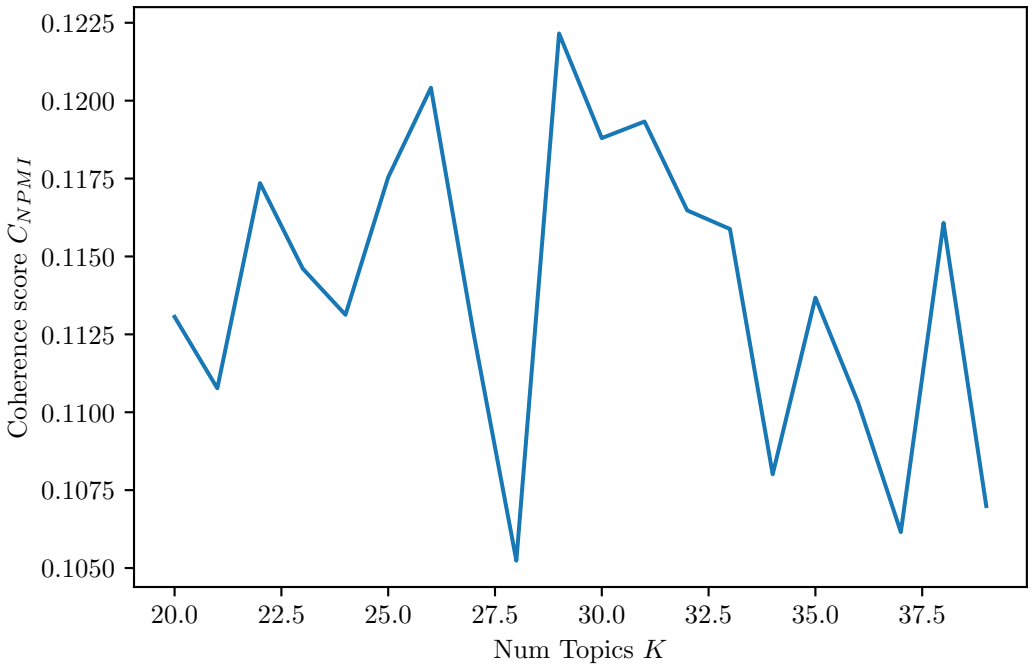
I collect WTI crude oil prices from the FRED Economic Database using the last closing price of the business week to represent weekly prices. The dates collected are from September 3rd, 2009 to January 18th, 2022. The main advantage of the WTI price is that it is available in a timely manner and not subject to data revisions (Alquist et al., 2013). In the literature, WTI has become the benchmark of global crude oil pricing due to US military and economic capabilities in the world (Liu and Huang, 2021).

4.2 Text Feature Extraction Results

Figure 8 shows graphically how NPMI coherence varies as I increase K number of topics. The search starts from $K = 20$, increasing by 1 every step, and ending at $K = 40$. At the end of every step, I calculate the NPMI coherence score. Graphically,

the search shows that the coherence is maximized at $K = 29$, with a coherence score of $0.122 \in [-1, 1]$ (closer to 1 means a stronger topic correlation). After $K = 29$, a steep downtrend is noticeable; more evident by a preliminary search (see Figure 11) of K topics with a wider range of K, specifically starting at $K = 5$ and ending at $K = 55$ with a larger step increase of 5. The coherence score drops severely after 30 topics.

Figure 8: NPMI Scores Varying K Topics



Furthermore, Table 2 reports the comparison of coherence scores between different methods of hyperparameter tuning that I explained in Section 3.2. In this table, I compare the effect of using lemmatization in the pre-processing step—the scores are significantly lower compared to the non-lemmatized methods. This is in line with a recommendation by Schofield et al. (2017) to not use lemmatization when processing text for topic models. The best performing LDA model was the Griffiths method without lemmatizing. The automatic version that Gensim offers shows scores that

are very close to Griffiths method in both cases, thus the Gensim version should not be overlooked when applying the processing step.

Table 3 reports all 29 latent topics along with the 8 most probable words belonging to a topic in descending order. How these topics vary over time can be seen in Figure 9 which displays the probability of the textual features in the article over the yearly time period. Figure 9 also displays the FinBERT sentiment results obtained and how the sentiment probabilities of the articles vary over time. The graphs labelled "pos", "neg", and "neutr" refer to the positive, negative and neutral probability for the articles aggregated weekly. For the positive graph, from the year 2009 to 2015, there is a downtrend in the positivity of articles. The trend continues upwards after 2015. The negativity of articles is a more consistent sentiment over time, remaining high throughout. For the neutral graph, articles have a high probability of having a neutral sentiment in 2009 then lowering around 2011. After 2011, the probabilities stay consistently low.

Next, I highlight some notable topics that can be captured by Table 3 and Figure 9. For the Organization of the Petroleum Exporting Countries (OPEC) related topic, the topic model splits the OPEC topic into three subtopics: Topic_1, Topic_19, and Topic_23. Topic_1 could be related to OPEC discussion on countries' petroleum output exportation. Topic_19 could be related to the OPEC forecast of barrels per day (BPD). Topic_23 seems related to OPEC and Russia's agreement in 2017 to extend oil production cuts. This can be confirmed by inspection of Figure 9 and the spike in the probability of Topic_23 around 2017. Topic_8 is extracting the topic about the European debt crisis (also known as the eurozone crisis) of February 2012. Topic_0 can be seen as the COVID-19 topic.

Table 2: Topic Coherence Scores

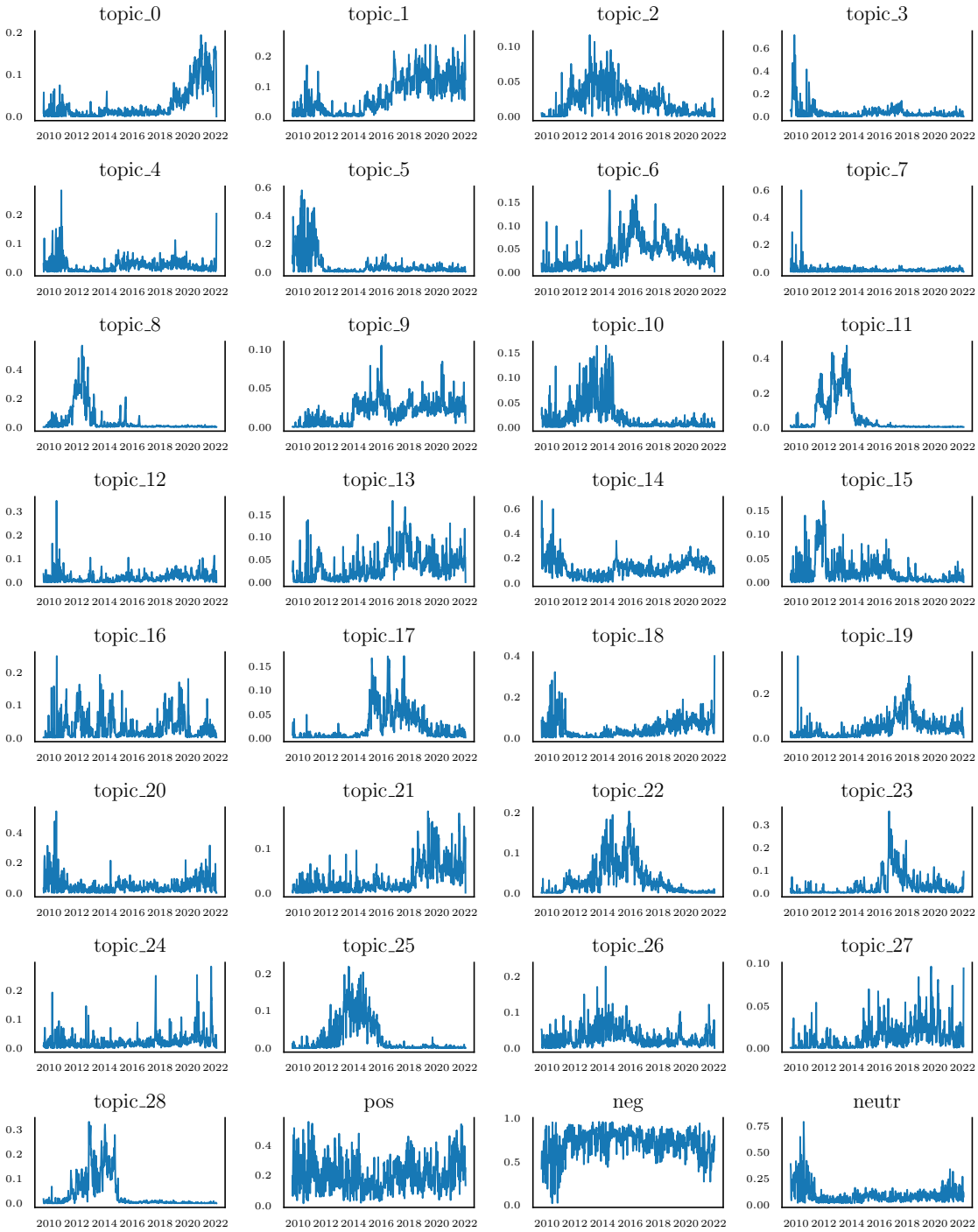
K	Griffiths (l)	Auto (l)	Auto	Griffiths
20	0.071	0.074	0.112	0.113
21	0.063	0.072	0.109	0.111
22	0.076	0.071	**0.121**	0.117
23	0.063	0.069	0.116	0.115
24	0.077	0.088	0.117	0.113
25	0.096	0.095	0.114	0.118
26	**0.106**	0.085	0.112	0.120
27	0.076	**0.101**	0.115	0.113
28	0.088	0.083	0.119	0.105
29	0.078	0.077	0.112	**0.122**
30	0.084	0.079	0.099	0.119
31	0.073	0.086	0.103	0.119
33	0.072	0.076	0.099	0.116
34	0.065	0.085	0.116	0.116
35	0.085	0.086	0.108	0.108
36	0.075	0.068	0.114	0.114
37	0.062	0.072	0.110	0.110
38	0.069	0.074	0.112	0.106
39	0.063	0.061	0.104	0.116
40	0.059	0.060	0.098	0.107

In Table 2, for column titles with "(l)" were topic models processed with lemmatization.

Table 3: Top 8 Words per Topic

Topic_0	Topic_1	Topic_2	Topic_3	Topic_4	Topic_5
covid	opec	release	stocks	china	percent
pandemic	output	tuesday	poor	venezuela	stocks
gold	countries	stockpiles	standard	india	points
benchmark	petroleum	later	fund	government	index
coronavirus	cuts	report	trading	korea	nasdaq
settled	organization	petroleum	etf	south	markets
per	exporting	weekly	sector	emerging	european
weekly	per	ahead	price	state	higher
Topic_6	**Topic_7**	**Topic_8**	**Topic_9**	**Topic_10**	**Topic_11**
us	gas	debt	texas	expectations	usd
trump	natural	euro	west	quarter	trade
level	natural_gas	zone	intermediate	sales	traded
iraq	resources	trade	west_texas_intermediate	expected	delivery
iran	renewable	european	contract	increase	exchange
delivery	settled_per_barrel	greece	benchmark	report	morning
june	companies	usd	month	showed	usd_barrel
amid	bbl	country	cushing	month	light
Topic_12	**Topic_13**	**Topic_14**	**Topic_15**	**Topic_16**	**Topic_17**
would	inventories	price	dollar	iran	friday
president	million_barrels	would	currencies	sanctions	rigs
administration	wednesday	even	greenback	iranian	baker
house	eia	one	basket	nuclear	rig
white	gasoline	time	major	east	hughes
white_house	information	much	six	middle	drilling
government	administration	years	commodities	tensions	weekly
state	stockpiles	like	expensive	tehran	count
Topic_18	**Topic_19**	**Topic_20**	**Topic_21**	**Topic_22**	**Topic_23**
fuel	bpd	company	china	delivery	opec
according	opec	billion	trade	inventories	russia
refiners	per	companies	world	exchange	producers
gasoline	output	earnings	variant	cents	output
sources	million_bpd	stock	chinese	trade	meeting
exports	shale	quarter	economic	report	deal
reuters	agency	share	trump	compared	cut
world	forecast	shares	growth	london	agreement
Topic_24	**Topic_25**	**Topic_26**	**Topic_27**	**Topic_28**	
gulf	manufacturing	fed	saudi	usd	
mexico	china	inflation	arabia	exchange	
gas	low	rate	saudi_arabia	delivery	
pipeline	month	federal	aramco	high	
hurricane	london	reserve	world	trading	
coast	level	bank	kingdom	york	
nyse	showed	central	saudis	session	
shale	york	friday	minister	articles	

Figure 9: Textual Features Overtime



In Figure 9, for all features, the Y-axis represents the probability, $[0, 1]$. For all the features with the label "topic", the graph is the weekly aggregated average probability of topic occurring in an article. The X-axis is yearly time, incrementing by 2 years

4.3 Comparing Forecasts

I compare OoS performances of the competing models: ARMA, ARMAX, Random Forest, and XGBoost using the forecasting method described in Section 3. I analyze 285 OoS performance metrics of one week ahead log price differences using 5 rolling windows, and a sub-sample size of 365. I use the last business day of the week to represent the week's overall price[11]. In this section, I answer the following questions: *How do the different forecasting methods perform? Does news help improve the prediction of crude oil prices? Which is the optimal forecasting model between competitors?*

In Table 4, if the row is labelled with *Text*, then it means that I use the forecasting method with textual features. If the row is labelled with *No Text*, then I forecast without the textual feature. To plot the performance metrics over time, Figure 10 displays the OoS performance metric of each sub-sample. Since the average of 5 rolling windows is taken to compute the performance metric in each sub-sample, I use the last rolling window day as the metric for that day. For instance, in sub-sample one, the performance metric plotted is the day for $t = 5$. For sub-sample two, the day for $t = 6$ is plotted, and so on. For the tree-based algorithms, I use one lag for price and for all textual features.

The following results answer the first question: *how do the different forecasting methods perform*? To note, for all competing models, they all fail to beat the rolling mean forecast. Henceforth, when comparing competing models, I exclude the rolling mean forecast except when I explicitly specify the rolling mean forecast.

For the average RMSE, between the competing models, they all perform relatively similar except for the Random Forest using no text features and the ARMAX model. The Random Forest with no text features has a noticeably higher RMSE than the other models while the ARMAX model has the lowest RMSE. When comparing using text versus no text within model types, the textual feature in the Random Forest

[11] I thank my supervisor, Dr. Ba Chu, for this suggestion

regression and ARMAX help improve the RMSE. Contrarily, textual features did not improve the RMSE for the XGBoost method. Figure 10 shows the RMSE values of the competing models for one period ahead. The XGBoost model has frequent spikes compared to the other competing models, which might signal slight overfitting.

Next, for the average MAE across competing models, the ARMAX regression with textual features had the lowest MAE. When comparing the use of text versus no text for individual models, textual features follow the same form as RMSE, where text improves the MAE in the Random Forest and the ARMA but worsens with the MAE for XGBoost. Figure 10 for MAE follows closely to the RMSE graph.

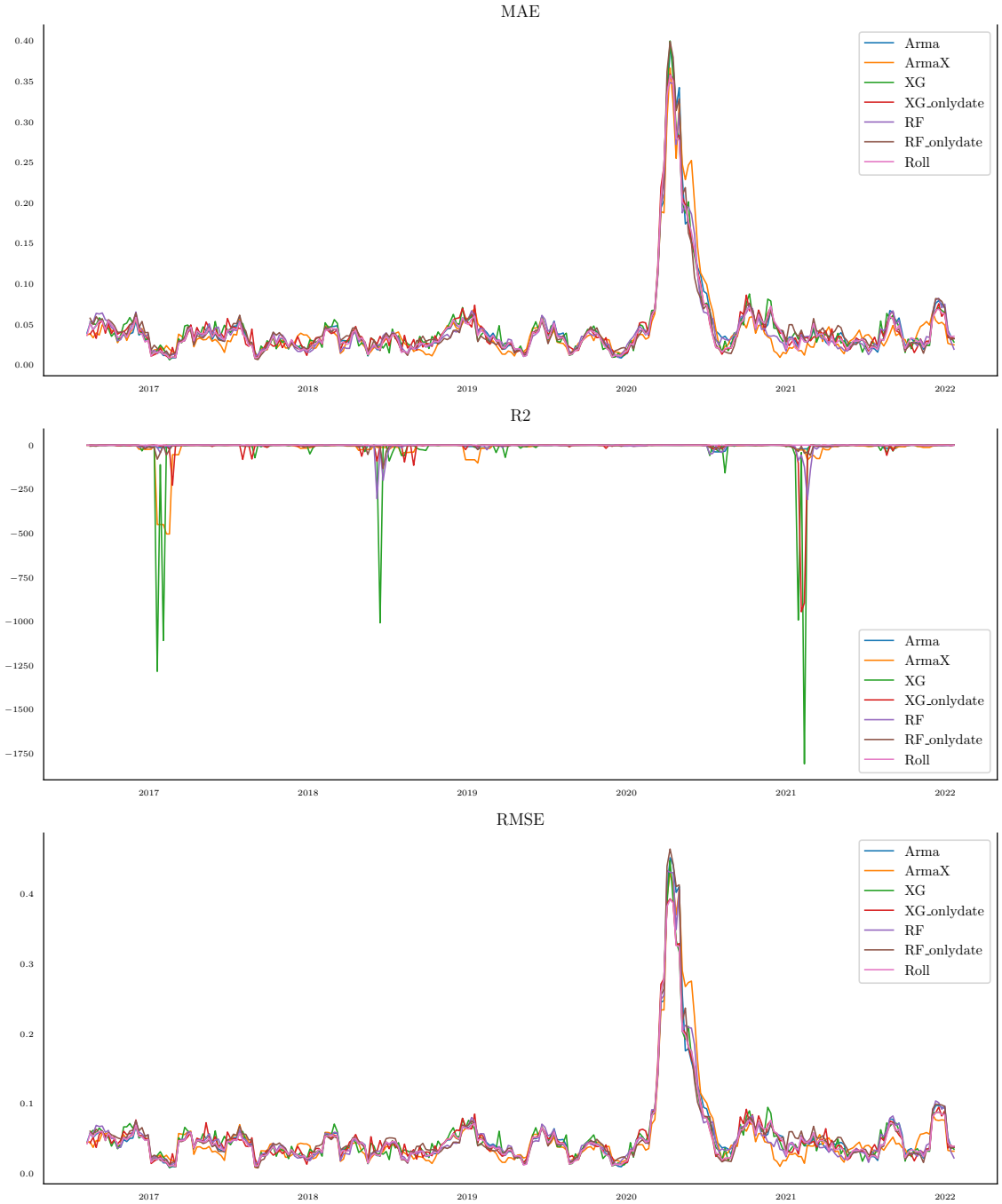
The average OoS R^2 values vary across models and methods. The R^2 has large spikes, so I remove them in Figure 12 to better see how the R^2 performs over time. In general, excluding the textual features performs the best in terms of R^2 and the ARMA model performs the best among all competing models. The XGBoost algorithms have the highest overall R^2 as well as the most dominant spikes, which is a sign of overfitting.

From the results I can draw the following conclusions:

1. *Does news help improve the prediction of crude oil prices*? The Random Forest and ARMAX algorithms perform better, which shows that adding textual features to the forecasting models improves the prediction. However, XGBoost did not improve with textual features.

2. *Which is the optimal forecasting model between competitors*? The optimal forecasting model among competing models is the ARMAX in terms of RMSE and MAE. The ARMA performed best in terms of R^2. However, the ARMAX and ARMA failed to beat the rolling mean forecast, thus the rolling mean forecast is still the best forecasting method.

Figure 10: Line plots of performance metrics for competing forecasts



In Figure 10, the Y-axis represents the scale of the corresponding metric. The X-axis is yearly time, starting from June, 15th 2016, ending January, 21st 2022. In the legend, models that end with "_onlydate" mean that the model was forecasted using no textual features.

Table 4: Forecasting results of one week ahead for crude oil prices

Forecasting Method	Features	RMSE	MAE	OoS R^2
Rolling Mean	-	0.05162	0.0352	0
ARMAX	*Text*	**0.05255**	**0.04320**	-16.75
ARMA	*No Text*	0.05353	0.04508	**-2.49**
XGBoost	*Text*	0.05376	0.04578	-27.78
	No Text	0.05319	0.04497	-11.48
Random Forest	*Text*	0.05378	0.04435	-7.11
	No Text	0.05446	0.04549	-4.08

5 Discussion

Since crude oil prices are mostly affected by demand factors, supply factors, or political developments (Filis et al., 2011), accurately forecasting crude oil prices can be difficult. I show that textual features can improve the competing forecasting models except for XGBoost. Notably, all models fail to beat the rolling mean forecast, thus I suggest that forecasters who pursue higher forecasting accuracy for oil prices add exogenous features such as variables from macroeconomic databases like FRED-MD and FRED-QD (McCracken and Ng, 2016, 2020). Forecasters could also benefit from adding leading indicators like crude oil inventories to enhance forecasting prediction, as news does not add sufficient enough accuracy alone.

An important event to note in Figure 10 is the impact of COVID-19 in April 2020, causing a crash in oil prices. This is clear from the large spike between the years 2020 and 2021 on the graph for RMSE and MAE. The XGBoost with no text and the rolling mean forecast minimize this error the best compared to the other models, while the rolling mean beat the XGBoost algorithm only slightly. The Random Forest with no textual features has the largest error of this crash. In terms of recovering from this spike in error, the ARMAX has the worst performance.

The textual features I obtain are similar to Bai et al. (2022), however, my framework uses state-of-the-art NLP models that are tuned specifically for the financial domain which makes my NLP models better suited for extracting textual features from crude oil news. The authors' framework differed from mine, as they did not use the content of the news articles and only the headlines. The motivation of their paper was a direct improvement on Li et al. (2019) as they recognized news headlines are noisy, making LDA a poor choice for topic modelling. As a result, Bai et al. used non-negative matrix factorization, which has its advantages such as better performance when dealing with shorter texts like headlines. However, my paper deals with long text, thus LDA is the correct choice for this experiment. Since my text mining framework is applied to different news sources than the previous authors mentioned, as well as using weekly rather than daily oil forecasts, a direct comparison of the performance metrics across studies is nonsensical. Nevertheless, I draw a similar conclusion as Bai et al. and Li et al. that adding textual features to the forecasting models can increase predictive forecasting performance, except for the XGBoost algorithm. In contrast to Bai et al. conclusion, instead found that adding textual features to all models except ARIMA improved forecasting accuracy.

This study has several strengths. First, I show using my framework that NLP can be a useful intermediary step when performing empirical analysis as seen in Section 4.2 where I demonstrate the usefulness of extracting topics over time and observing the most probable words in Table 3 simultaneously. This gives the ability to observe in-depth knowledge of news information purely in an unsupervised fashion without the need for a labelled data set. Second, I make use of state-of-the-art NLP models to extract the textual features instead of lexicon (dictionary) based methods which have several limitations as discussed in Section 1.1. Lastly, my framework does not make any theoretical assumptions that require expert opinion (Chen, 2021; Ke et al., 2019; Luss and d'Aspremont, 2015) as our task of extracting textual features is purely

unsupervised.

The study also has several limitations. First, the textual features that I extract come from two sources which often have the same set of authors writing about crude oil. This may lead to bias in the textual data set; the data set is likely biased by author opinions. More specifically, the textual features are more positive or negative depending on the author's attitudes towards the topic. Second, I remove articles that contained less than 5 of the keyword "oil", which is a harsh way to remove articles. Future work could benefit from exploring better ways of removing articles. A suggestion is to perform topic modelling on the set of data with these articles, then keep articles that the topic model deems related to oil. Third, while the novel forecasting strategy I use has its advantages, it comes with a disadvantage: being computationally expensive. This limits the selection of forecasting models I use and their lags. Fourth, for sentiment analysis, FinBERT has its limitation: being that it is an "off-the-shelf" black-box algorithm. It becomes difficult to assess the accuracy of unlabelled data sets and unsupervised tasks. Although noting FinBERT's assessment difficulties of performance, the extensive research on FinBERT's track-record performance (Araci, 2019; Mishev et al., 2020) should not be disregarded as it achieves state-of-the-art on virtually every financial domain-specific language understanding task. Since this study is on unsupervised ways of dealing with textual data, labelling the data set is beyond the scope of the study. Fifth, as I discussed in Section 3.3, my method of addressing the token limit size assumes that each chunk has equal sentiment information about the document. Future work could explore different ways of addressing BERT's 512 token limit such as the "Long Transformer" (Beltagy et al., 2020).

In future work, Dynamic Topic modelling (Blei and Lafferty, 2006) could be explored as another way to extract latent topics over time. This method of extracting latent topics works better for a collection of documents that evolve over time. For example, Topic_8 from Table 3 and Figure 9 is extracting the topic about the European

debt crisis (also known as the Eurozone Crisis) of February 2012. However, as time moves past 2012 this topic is no longer probable as the event has passed and should no longer be considered a topic in the years after 2012. Dynamic topic modelling can highlight situations like this better than the static LDA topic model that I use in this study. Another method that could be explored to tackle this problem is a feature selection method for each sub-sample. This might filter out topics that are not important for the time period.

6 Conclusion

I propose and analyze a new framework for forecasting crude oil using state-of-the-art text mining techniques and construct a horse-race among economic and machine learning models. I enhance text mining methods that were once used for obtaining textual features from news articles and further improve it by using FinBERT to generate contextual embedding to obtain sentiment probabilities and using LDA to identify latent topics in the news articles. The main take-aways from this forecasting experiment are: (1) textual features extracted from news text using my framework can add predictive power to the forecasts; (2) news on its own is not enough to forecast oil prices at a higher accuracy; (3) morphological transformation methods of text harm topic coherence scores and should not be used in the processing step when performing topic modelling.

References

Alquist, Ron, Lutz Kilian, and Robert J Vigfusson (2013) 'Forecasting the price of oil.' In 'Handbook of economic forecasting,' vol. 2 (Elsevier) pp. 427–507

Araci, Dogu (2019) 'Finbert: Financial sentiment analysis with pre-trained language models.' *arXiv preprint arXiv:1908.10063*

Asuncion, Arthur, Max Welling, Padhraic Smyth, and Yee Whye Teh (2012) 'On smoothing and inference for topic models.' *arXiv preprint arXiv:1205.2662*

Bai, Yun, Xixi Li, Hao Yu, and Suling Jia (2022) 'Crude oil price forecasting incorporating news text.' *International Journal of Forecasting* 38(1), 367–383

Beltagy, Iz, Matthew E Peters, and Arman Cohan (2020) 'Longformer: The long-document transformer.' *arXiv preprint arXiv:2004.05150*

Bird, Steven, Ewan Klein, and Edward Loper (2009) *Natural language processing with Python: analyzing text with the natural language toolkit* (" O'Reilly Media, Inc.")

Blei, David M., and John D. Lafferty (2006) 'Dynamic topic models.' *Proceedings of the 23rd international conference on Machine learning*

Blei, David M, and Michael I Jordan (2004) 'Variational methods for the dirichlet process.' In 'Proceedings of the twenty-first international conference on Machine learning' p. 12

Blei, David M, Andrew Y Ng, and Michael I Jordan (2003) 'Latent dirichlet allocation.' *the Journal of machine Learning research* 3, 993–1022

Bouma, Gerlof (2009) 'Normalized (pointwise) mutual information in collocation extraction.' *Proceedings of GSCL* 30, 31–40

Breiman, Leo (2001) 'Random forests.' *Machine learning* 45(1), 5–32

Chang, Jonathan, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei (2009) 'Reading tea leaves: How humans interpret topic models.' In 'Advances in neural information processing systems' pp. 288–296

Chen, Qinkai (2021) 'Stock movement prediction with financial news using contextualized embedding from bert.' *arXiv preprint arXiv:2107.08721*

Chen, Tianqi, and Carlos Guestrin (2016) 'Xgboost: A scalable tree boosting system.' In 'Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining' pp. 785–794

Chu, Ba, and Shafiullah Qureshi (2021) 'Comparing out-of-sample performance of machine learning methods to forecast u.s. gdp growth'

Cowles, Alfred (1933) 'Can stock market forecasters forecast?' *Econometrica: Journal of the Econometric Society* pp. 309–324

Deerwester, Scott C., Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard A. Harshman (1990) 'Indexing by latent semantic analysis.' *Journal of the Association for Information Science and Technology* 41, 391–407

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019) 'Bert: Pre-training of deep bidirectional transformers for language understanding'

Ek, Adam, Jean-Philippe Bernardy, and Stergios Chatzikyriakidis (2020) 'How does punctuation affect neural models in natural language inference.' In 'Proceedings of the Probability and Meaning Conference (PaM 2020)' pp. 109–116

Ellingsen, Jon, Vegard Larsen, and Leif Anders Thorsrud (2020) 'News media vs. fred-md for macroeconomic forecasting'

Filis, George, Stavros Degiannakis, and Christos Floros (2011) 'Dynamic correlation

between stock market and oil prices: The case of oil-importing and oil-exporting countries.' *International review of financial analysis* 20(3), 152–164

Goldberg, Yoav, and Omer Levy (2014) 'word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method.' *arXiv preprint arXiv:1402.3722*

Griffiths, Thomas L, and Mark Steyvers (2004) 'Finding scientific topics.' *Proceedings of the National academy of Sciences* 101(suppl 1), 5228–5235

Hamilton, James D (2009) 'Causes and consequences of the oil shock of 2007-08.' Technical Report, National Bureau of Economic Research

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009) 'Boosting and additive trees.' In 'The elements of statistical learning' (Springer) pp. 337–387

Hofmann, Thomas (1999) 'Probabilistic latent semantic indexing.' In 'Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval' pp. 50–57

Ke, Zheng Tracy, Bryan T Kelly, and Dacheng Xiu (2019) 'Predicting returns with text data.' Technical Report, National Bureau of Economic Research

Kilian, Lutz, and Daniel P Murphy (2014) 'The role of inventories and speculative trading in the global market for crude oil.' *Journal of Applied econometrics* 29(3), 454–478

Lacoste-Julien, Simon, Fei Sha, and Michael Jordan (2008) 'Disclda: Discriminative learning for dimensionality reduction and classification.' *Advances in neural information processing systems*

Larsen, Vegard H, and Leif A Thorsrud (2019) 'The value of news for economic developments.' *Journal of Econometrics* 210(1), 203–218

Lee, Jean, Hoyoul Luis Youn, Nicholas Stevens, Josiah Poon, and Soyeon Caren Han (2021) 'Fednlp: An interpretable nlp system to decode federal reserve communications.' In 'Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval' pp. 2560–2564

Li, Xuerong, Wei Shang, and Shouyang Wang (2019) 'Text-based crude oil price forecasting: A deep learning approach.' *International Journal of Forecasting* 35(4), 1548–1560

Liu, Jiangwei, and Xiaohong Huang (2021) 'Forecasting crude oil price using event extraction.' *IEEE Access* 9, 149067–149076

Loughran, Tim, and Bill McDonald (2011) 'When is a liability not a liability? textual analysis, dictionaries, and 10-ks.' *The Journal of finance* 66(1), 35–65

Luss, Ronny, and Alexandre d'Aspremont (2015) 'Predicting abnormal returns from news using text classification.' *Quantitative Finance* 15(6), 999–1012

Mcauliffe, Jon, and David Blei (2007) 'Supervised topic models.' *Advances in neural information processing systems*

McCracken, Michael, and Serena Ng (2020) 'Fred-qd: A quarterly database for macroeconomic research.' Technical Report, National Bureau of Economic Research

McCracken, Michael W, and Serena Ng (2016) 'Fred-md: A monthly database for macroeconomic research.' *Journal of Business & Economic Statistics* 34(4), 574–589

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013a) 'Distributed representations of words and phrases and their compositionality.' *Advances in neural information processing systems*

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013b) 'Efficient estimation of word representations in vector space.' *arXiv preprint arXiv:1301.3781*

Minka, Thomas P, and John Lafferty (2012) 'Expectation-propogation for the generative aspect model.' *arXiv preprint arXiv:1301.0588*

Mishev, Kostadin, Ana Gjorgjevikj, Irena Vodenska, Lubomir T Chitkushev, and Dimitar Trajanov (2020) 'Evaluation of sentiment analysis in finance: from lexicons to transformers.' *IEEE access* 8, 131662–131682

Mohan, Saloni, Sahitya Mullapudi, Sudheer Sammeta, Parag Vijayvergia, and David C Anastasiu (2019) 'Stock price prediction using news sentiment analysis.' In '2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)' IEEE pp. 205–208

Mukherjee, Indraneel, and David M Blei (2009) 'Relative performance guarantees for approximate inference in latent dirichlet allocation.' In 'Advances in Neural Information Processing Systems' pp. 1129–1136

Newman, David, Jey Han Lau, Karl Grieser, and Timothy Baldwin (2010) 'Automatic evaluation of topic coherence.' In 'Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics' pp. 100–108

Pappagari, Raghavendra, Piotr Żelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak (2019) 'Hierarchical transformers for long document classification'

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

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (2018) 'Deep contextualized word representations.' *CoRR*

Porteous, Ian, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling (2008) 'Fast collapsed gibbs sampling for latent dirichlet allocation.' In 'Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining' pp. 569–577

Rehurek, Radim, and Petr Sojka (2011) 'Gensim–python framework for vector space modelling.' *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*

Röder, Michael, Andreas Both, and Alexander Hinneburg (2015) 'Exploring the space of topic coherence measures.' In 'Proceedings of the eighth ACM international conference on Web search and data mining' pp. 399–408

Schofield, Alexandra, Måns Magnusson, Laure Thompson, and David Mimno (2017) 'Understanding text pre-processing for latent dirichlet allocation.' In 'Proceedings of the 15th conference of the European chapter of the Association for Computational Linguistics,' vol. 2 pp. 432–436

Sonkiya, Priyank, Vikas Bajpai, and Anukriti Bansal (2021) 'Stock price prediction using bert and gan.' *arXiv preprint arXiv:2107.09055*

Špeh, Jaka, Andrej Muhič, and Jan Rupnik (2013) 'Algorithms of the lda model [report].' *arXiv preprint arXiv:1307.0317*

Teh, Yee, David Newman, and Max Welling (2006) 'A collapsed variational bayesian inference algorithm for latent dirichlet allocation.' *Advances in neural information processing systems*

Tetlock, Paul C (2007) 'Giving content to investor sentiment: The role of media in the stock market.' *The Journal of finance* 62(3), 1139–1168

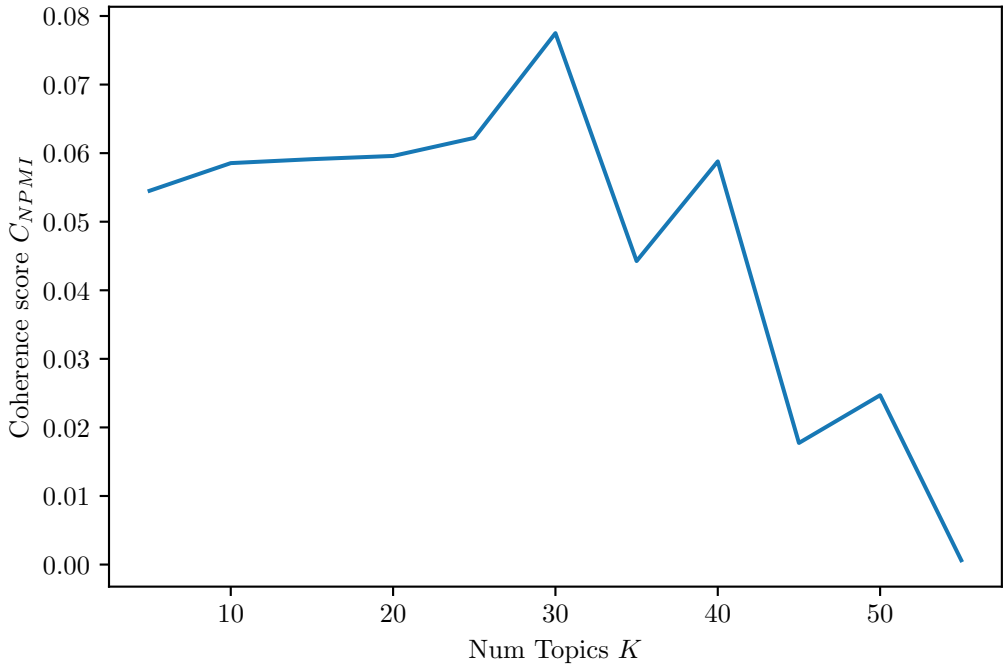
Wallach, Hanna M, Iain Murray, Ruslan Salakhutdinov, and David Mimno (2009) 'Evaluation methods for topic models.' In 'Proceedings of the 26th annual international conference on machine learning' pp. 1105–1112

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush (2020) 'Huggingface's transformers: State-of-the-art natural language processing'

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean (2016) 'Google's neural machine translation system: Bridging the gap between human and machine translation.' *CoRR*

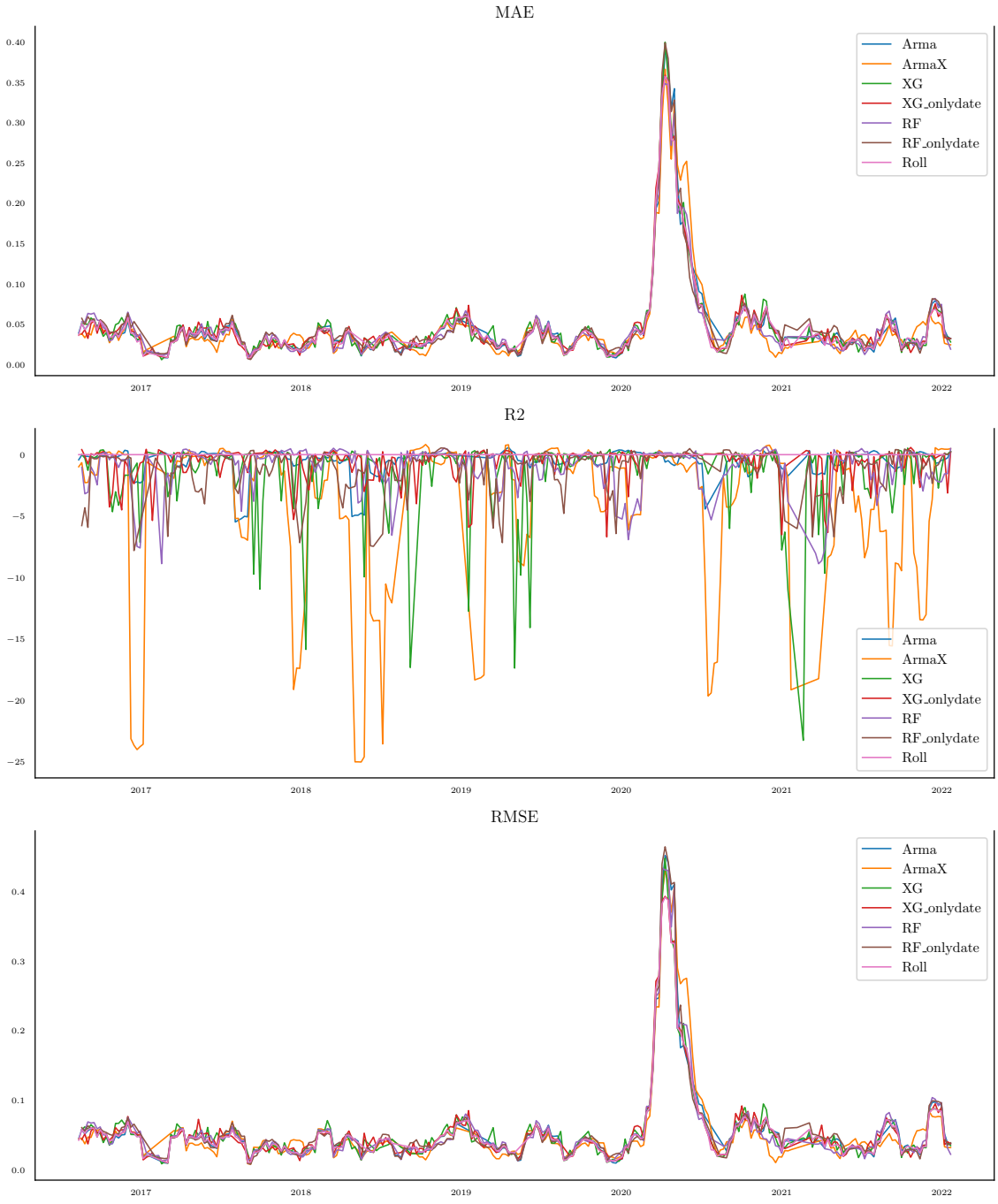
A Additional Figures

Figure 11: Preliminary Search C_{NPMI} of Varying K Topics



In Figure 11, K topics range from $[5, 55]$ with a 5 step increase at every C_{NPMI} evaluation. The parameters used were from the Griffiths method of fixing α and β while varying K topics.

Figure 12: Line plots of performance metrics for competing forecasts with spikes removed



In Figure 12, the Y-axis represents the scale of the corresponding metric. The X-axis is yearly time, starting from June, 15th 2016, ending January, 21st 2022. In the legend, models that end with "_onlydate" mean that it was forecasted using no textual features. How this plot differs from Figure 10, is that the outliers in the R^2 are removed using the 10% quantile and removing it.

B Additional Material

B.1 Hyperparameters of Random Forest 2.5

First, I explain each hyperparameter briefly for the Random Forest algorithm, then show the grid parameters I use in the walk-forward validation.

The Random Forest hyperparameters are:

1. Maximum depth: puts a limit on the depth of each tree built. This is determined by the longest path from the root node to the leaf node. I set the parameter grid for this value to be between $[10, 110]$, with a step of 10.

2. Minimum sample split: this is the minimum number of samples required to split an internal node. I set the parameter grid for this value to be $[2, 5, 8]$.

3. Maximum terminal (leaf) nodes: this puts a limit on how many terminal nodes a tree can have. This restricts tree growth and could lead to underfitting if this value is too small. I leave this parameter to its default value which allows for unlimited terminal nodes.

4. Minimum samples per terminal node: this is the minimum number of samples required to be at a terminal node. If the terminal node does not meet this requirement, the split will not happen. I set the parameter grid for this value to be $[1, 2, \ldots, 6]$.

5. Number of tree estimators: this is the number of trees built. I set the parameter grid for this value to be between $[50, 350]$ with a step of 50.

6. Maximum samples per tree: the number of samples to draw from data to train each tree. This is left to "None", and draws an equal amount of data to each sub-sample.

7. Maximum features per tree: this puts a limit on the number of features given to each tree. However, for regression trees, it is common to set this value to consider all features in the data set.

B.2 Hyperparameters of XGBoost

First, I explain each hyperparameter briefly for the XGBoost algorithm (unless it overlaps with B.1), then show the grid parameters I use in the walk-forward validation. Note that I list only the parameters used for my walk-forward validation, the XGBoost algorithm has many more parameters, refer to the documentation for a detailed list of parameters[12]. The XGBoost hyperparameters are:

1. Learning rate (η): The shrinkage weight that scales the contribution of each tree by a factor $\eta \in (0, 1)$. I set the parameter grid for this value to be between $[0.01, 0.1]$ with a step of 0.01.

2. L1 regularization(α): I set the parameter grid for this value to be between $[0, 50]$ with a step of 10.

3. L2 regularization (λ): I set the parameter grid for this value to be between $[0, 100]$ with a step of 20.

4. Minimum loss reduction for split (γ): Sets a limit on every node, where the node will only split if the leaf nodes reach a minimum loss reduction. I set the parameter grid for this parameter to be between $[0, 0.8]$ with a step of 0.01.

5. Minimum child weight: Sets a limit on the sum of weights of all observations required in a child. If the leaf node sum is less than the limit, the tree-building process stops. I set the parameter grid for this parameter to be $[2, 3, 4, 5]$.

[12] XGBoost Documentation

6. Sampling ratio of columns: sub-sample ratio of columns when constructing each tree. I set the parameter grid for this value to be between $[0.1, 0.9]$ with a step of 0.1.

7. Sub-sample: Fraction of observations to be randomly sampled for each tree. I set the parameter grid for this parameter to be $[0.7, 0.8, 0.9]$.

8. Maximum depth (see B.1): I set the parameter grid for this parameter to be $[1, 2, \ldots, 6]$.

9. Number of tree estimators (see B.1): I set the parameter grid for this parameter to be $[100, 500, 1000]$.

B.3 Dynamic Chunking Aggregation Method of BERT

The first step in BERT is to tokenize the documents.[13] However, tokenization takes more of a unique approach in BERT than simply splitting text into word tokens. BERT requires a fixed-length input of 512 tokens. If the length of the sequence is shorter than 512 tokens, the [PAD] tokens are appended to the end of the sequence until it reaches the 512 token length. BERT then gives every single token an ID based on the sub-word segment tokenization algorithm, WordPiece (Wu et al., 2016). This algorithm has a simple but ingenious way of dealing with out-of-vocabulary words by breaking the word down into its sub-word, or even character-word level. For example, the word "embeddings" is not a whole word in the WordPiece vocabulary, but is broken down into sub-tokens: ["em", "##bed", "##ding", "##s"]. Next, it is then transformed into its token ID's: [7861, 8270, 4667, 2015]. The latter is inputted into BERT's input layer. WordPiece has 30,000 token embeddings to look up the token associated with the word.

Since the majority of articles in the data set is well over the 512 token input limit,

[13] I use the HuggingFace transformers library (Wolf et al., 2020)

it is necessary to split the input tokens into even M_d sized chunks for each document with a maximum token length of 510 (since I add [CLS] and [SEP] tokens at the end of the token input sequence, it equals 512). Let N_d be the token length for document d. I then dynamically determine the chunk size for input tokens sequence for each document:

$$M_d = \lceil \frac{N_d}{\lceil \frac{N_d}{510} \rceil} \rceil \tag{17}$$

Next, I convert the whole document into its respective WordPiece ID tokens then add the [CLS] to the beginning of each token input sequence chunk and [SEP] at the end. If $M_d < 512$ tokens, I append $512 - M_d$ [PAD] tokens at the end of the [SEP] token for each sequence chunk until the chunk reaches its 512 token length[14]. Since BERT uses the attention mechanism, it requires an attention mask input $m_{d,n} \in \{0, 1\}$. This input must follow the same requirement length for BERT. $m_{d,n}$ maps to each respective input token in the sequence, therefore, it follows the same chunking process as the input tokens. If the mask token is 1, it tells BERT to apply the attention mechanism to this token, if the mask token is 0, do not apply the attention mechanism to this token. It is common practice that every input token, other than the [PAD] token needs the attention mechanism. Therefore, $m_{d,n} = 0$ for all $512 - M_d$ and $m_{d,n} = 1$ otherwise.

[14]When describing the algorithmic process, I refer to the special tokens as its non-ID form: [PAD], [CLS], and [SEP]. However, the transformation is done by using their token ID's: 0, 101, and 102 respectively. I explain the process in its non-ID form to ease the understanding