Commodities Forecasting from Non-Financial World News from GDELT

Abstract

We propose a new method for predicting commodity price data through the use of aggregated daily non-macroeconomic news. Our method creates a static clustering to extract topics and weighs each day's news events with an extracted importance metric. We use the GDELT news database to extract raw news attributes and sentiment evaluations of the articles. With a K-means model for clustering news data feeding into a lasso regression model we predict the log returns of several commodity prices, finding the most success on silver, where we achieved a 0.65 binary accuracy rate and a 0.37 mean absolute error. When using a Kelly criterion based trading strategy, with training on 563 days and testing on 221, we lose 0.6% of the investment. The S&P fell by 1.2% in this time. By significantly sparsifying the ARIMA model, we made 0.7% returns.

1. Introduction

Financial data prediction is appealing both for its challenging nature and for its practical applications. Because market dynamics are complex and random, accurate price prediction is difficult. There are two types of market prediction techniques: fundamental analysis, which relies on an asset's data for forecasting, and technical analysis, which relies on historical trends to exploit market timing (Schumaker & Chen, 2009). This paper examines the use of news data to augment technical analysis for prediction of commodity prices.

We focus on a comprehensive set of non-economic news data unrelated to financial markets, interest rates, stock prices, and the like. In particular, we will analyze whether a summary of a day's news topics is predictive of commodity prices the next day. To avoid losing information about the significance of singular events, we will rely on the sentiment analysis already applied to our dataset to calculate how each news event should be weighted in a daily news summary.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1.1. Commodity Prediction

We choose to predict commodities prices because while various studies have analyzed the effect of news on stock prices (McQueen & Roley, 1993) and foreign exchange rates (Kamruzzaman et al., 2003), relatively little work has looked at applying news data to the prediction of the similar commodities market. We use non-economic news, which does not have in-depth numeric data, because the influence of economic news has already been examined in the past by many researchers (Gidófalvi & Elkan, 2001)(Schumaker & Chen, 2009)(Bollen et al., 2011)(Hagenau et al., 2012). We believe real-time news information has predictive power for commodity prices, because, since commodities, by definition, must be extracted or produced by countries, it is likely that underlying factors of their production rates will be captured by local news, especially reporting on crisis events. We assume there is a set of news basis vectors that is stable over time. Prices are also sensitive to current conditions because the supply and demands that drive them are inelastic, or unaffected by changes in price (Chen et al., 2008).

1.2. GDELT Dataset

We draw news from the Global Database of Events, Language, and Tone (The GDELT Project, a), which aggregates news from broadcast, print, and web sources across 100 languages and parses out features to describe each news event. Each news event has 58 features which can be broadly divided into either topical or importance-related data. Topical data might include the two actors involved in an event, the general sentiment of the language used (extracted using 18 content analysis systems), the location of actors, or broad categorization of the type of event occurring (trade agreement, violent action, etc) in categorical codes (CAMEO codes). Importance-related features deal with the magnitude of the event and are drawn from metrics such as the number of news sources mentioning the event, or the total number of articles written about it. GDELT is available for free and uses a variety of international news sources with daily updates, and contains more than 250 million events from 1979 to present, or roughly 100,000 per day. GDELT's predictive potential for financial markets has been positively assessed by researchers who used GDELT to examine the impacts of the June 2013 Southeast Asian heat wave and the December 2013 Indian riots and the effect they had on the Singapore stock market (Phua

et al., 2014).

2. Motivating Analysis

2.1. GDELT Dataset

2.1.1. Sparsity in GDELT

The space of news events spanned by all columns in GDELT is much larger than the subspace we expect news to lie on. There are likely to be at least two modes of low-dimensional interactions in the data: (1) that actors only interact within small cliques and (2) that each actor is involved in a small set of events. We conducted an initial analysis on a random sample of days before August 2015 to avoid making conclusions that overfit the test data.

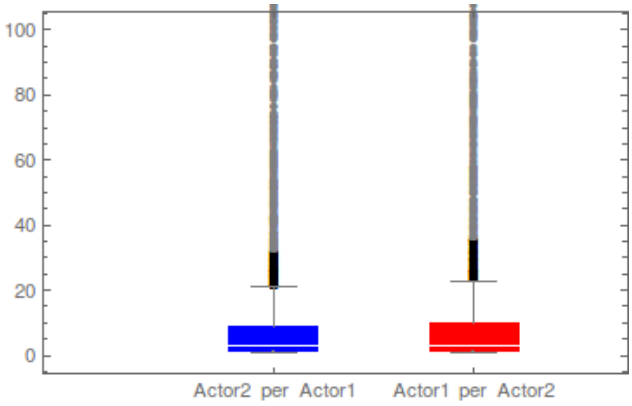

Figure 1. Box-and-whisker plot of number of distinct actors each CAMEO actor interacts with (each event may have up to 2 involved actors, where the first inflicts the action), performed on the day-stratified random sample of the events. The medium number of co-actors for both Actor1 and Actor2 categories is 3, with a Q3 of 9 and 10 and maximum of 988 and 990, respectively. The sample contained about 124K events total. The outliers with many interactions are generic or common names, such as `PRESIDENT` or `UNITED STATES`.

As Figure 2.1.1 demonstrates, actor count is a heavily skewed distribution. This gives us confidence that actors are indeed in small cliques for the sampled days. We conduct a similar inspection for the number of unique CAMEO coded events per actor in Figure 2.1.1:

Because of the sparsity that is present, we wish to reduce the dimension of our data, which originally consists of over 300 million points in a space spanned by 58 numeric, categorical, and string columns for three reasons: 1) to more accurately represent it, 2) so we can generate models in a continuous space of reduced-dimension tuples of real values (instead of having some categorical values), and 3) so our data pipeline only has to handle a reduced data size.

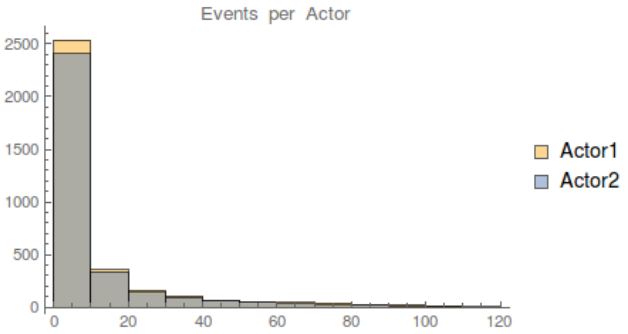

Figure 2. Number of events that occur for individual actors. The median actor encounters 3 events, and the 75% most active ones still see less than 15. This diagram only shows the 95% least active actors.

We found that classical dimensionality reduction was not tractable to apply to a dataset of this size - the highly categorical nature of the dataset results in a large dimensional expansion when preparing numeric inputs to the algorithms. Because of this, and because of our dataset's observed sparsity, we turned to a clustering based approach.

3. Related Work

On the whole, predicting any financial market has proven to be difficult. Simon's work encapsulates many of the inherent problems. Price predictions might accurately predict directionality changes or general trends, but if these predictions consistently lag behind actual prices, they will not be useful despite a low RMSE. Model evaluation therefore must either demonstrate ability to generate profit through a trading scheme or demonstrate an ability to correctly predict directionality for desired trading period times and not merely a low RMSE. Simon identifies a training period of 3 years and a test period of 6 months as optimal for reducing predicted errors (Simon, 2002).

In an examination of the potential for technical approaches to predict pricing movements using a Naive Bayes classifier, Gidofalvi found that a 20 minute period before and after the release of financial news allowed for a weak prediction of price movements (Gidófalvi & Elkan, 2001). It was found by McQueen and Roley that fundamental macroeconomic news has little impact on stock prices but that other news types have effects dependent on responses of expected flows relative to equity discount rates (McQueen & Roley, 1993).

Using textual analysis and SVMs for prediction, Schumaker and Chen were able to demonstrate 57% directional accuracy when using breaking financial news articles to

predict S&P500 stock movements within 20 minute periods (Schumaker & Chen, 2009). Bollen et al also achieved an accuracy of 86.7% when incorporating semantic mood data from Twitter and using self-organizing fuzzy neural nets (Bollen et al., 2011).

A new area for price prediction is outlined by Phua et al, who validate the mostly non-financial GDELT as a valid news source for market prediction. This is in contrast to the financial data sets used by Schumaker and Chen, which is comprised of USecurity and Exchange Commission reports, stock related information such as from The Motley Fool, buy/sell/hold recommendations, and other similar news. GDELT does not have the numeric economic data we wish to avoid. Phua et al's topic analysis shows that GDELT consistently identifies impactful events on stock markets, despite its lack of financial focus. Term extraction from news sources is shown to be relevant using concept link exploration. However, Phua et al do find that not all significant events can be distinguished by GDELT. For example, the first riots in Singapore in 40 years did not appear significantly different than other news clusters. They were also unable to verify the quality of semantic score assignments. While they use decision trees to determine the factors potentially useful in price forecasting, they do not attempt to actually forecast prices in contrast to this paper.

4. Methods

Recent GDELT updates provide feeds with 15-minute resolution (The GDELT Project, b), but historical feeds offered only daily resolution. We decided for simplicity to analyze daily changes in price, taking a full day's worth of news data into account. This was most in line with testing our hypothesis, that an aggregation of topic clusters in a day's news is predictive of commodity pricing.

Our data pipeline was designed to provide day-wise parallelism over our data set in order to enable fast testing of new feature extraction methods. The raw data, as well as intermediate data, is stored in TSV ASCII format.

4.1. Model

Every news event in GDELT is stored as a row in a file for that day's report. Every set of rows may undergo a series of transformations. First, in the **preprocessing** stage, we extract relevant topic- and importance- related columns. For topics, a mix of numeric and categorical fields are extracted. Importance columns are saved for use in a further step. Next, in the **expansion** stage, we project each row of topic features to a purely real space by using one-hot encoding for categorical values. For each category, one-hot encoding produces n dummy boolean variables, where n is the number of unique categories. Finally, in the **summary** stage, K-means clustering is performed on this purely numeric representation of the data. Euclidean distance is used as the distance metric for K-means. Many clustering models are created with a range of K from 10 to 5000. We use K-means for tractibility reasons.

The day summaries are then conglomerated in a time series $\{\mathbf{d}_i\}$. For a day-indexed time series of a commodity's price, $\{p_i\}$, we have the corresponding sequence of binary labels indicating whether price has increased the next day, $y_i = 1_{p_{i+1}>p_i}$.

Thus, our model assumes that:

1. The static clustering of news topics is accurate for the future.
2. $\mathbb{P}(p_{i+1} > p_i)$ may be determined by a regression over summaries $\mathbf{d}_i$.

Both of these are strong assumptions. The first may be weakened by adapting a dynamic clustering model, such as an infinite Gaussian Mixture Model or a Hierarchical Dirichlet Process. The second requires $\mathbf{d}_i$ to be both contain a sufficient amount of information to predict price behavior, which is a matter of appropriate feature extraction, and the assumption that we do not lose too much predictive power in summarizing a day by aggregating over the clusters to which its news events belonged.

4.2. Execution

Our current implementation uses K-means for clustering and logistic regression for classification. We trained, validated, and sampled from the days between 2006-01-01 and 2015-07-31. Recent days were used for testing.

For tractability reasons, we use a random sample of one million events to create the clustering models which are used in the main pipeline. This sample goes through the same preprocessing and expansion pipeline as the day summaries which are used in the regression except its output is used to create a clustering model. The sample is drawn uniformly from all events without any recency bias.

The raw data starts with $X = 58$ columns. Each day has around $N = 100,000$ rows, but this is variable between days. The total size of the set is about 60GB.

The preprocessing step does standard data cleaning such as removing malformed rows but more importantly removes all but 19 of the original columns, and also groups the remaining into $T = 12$ topic-related columns and $I = 9$ importance columns. The removed columns have either redundant data or data deemed insufficiently relevant to commodity prediction. The topic columns are in a compressed format, with categorical variables represented as integers. String columns are sanitized and have stop words removed.

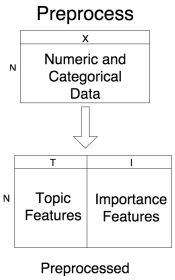

Figure 3. Preprocessing stage

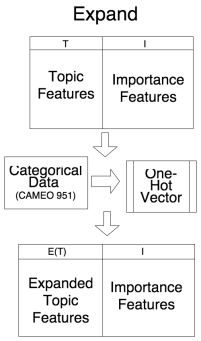

Figure 4. Expansion stage

The expansion step one-hot encodes categorical features, such as the event CAMEO codes, producing n dummy boolean variables, where n is the number of unique categories. If the data for a categorical column is missing, its corresponding one-hot vector is the zero vector. The 19 feature columns are expanded to 938.

As mentioned, the random sample is used to generate K-means models. These models are then used to cluster each day's data. We extract the NxK clustered data and separate it into two matrices containing clustered topic data and clustered importance data and pad these matrices with 1s, as in 4.2. We then multiply the tranposed topic and importance matrices together. Because of the padded 1s, their multiplication conveniently extracts the sum of importance-weighted events belonging to certain topics. Note that this model is extensible to a mixture model, where each event's contribution to a topic's importance for that day is weighed by the probability of belonging to that topic's class. Furthermore, the aggregation accomplished by the multiplication creates a highly reduced and uniform description of every day which is a flattened vector of dimension dependent on the cluster count.

These daily summaries are then enriched with historical pricing data before being fed into the linear model. The

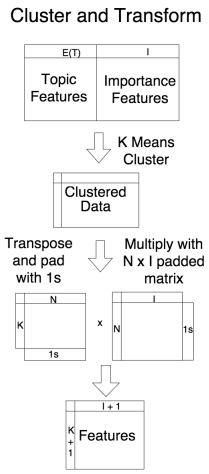

Figure 5. Summary stage

pricing data added are the 5, 10, and 30 day rolling averages of the commodity price.

4.2.1. Autoregressive Models: ARMA

We started our exploratory analysis with binary classification. We used diverse classification techniques such as Logistic Regression and Support Vector Machine on the sign of the change of the returns from a day to the next. The prices of silver from 2013 to 2015 were split between training, validation and testing in a 60:20:20 ratio. The best parameters from each model were chosen based on the validation accuracy. However, our test accuracy results were sub-par: Logistic Regression achieved a binary classification accuracy of 54.7% and SVM achieved a binary classfication accuracy of 58.6% . These models barely exceeded the baseline achieved by always guessing positive labels on the test set which would produce a 54% accuracy. Therefore, we decided to employ Autoregressive models which combine the news data with the time series data in order to produce continuous predictions of next-day time series values.

Autoregressive models are some of the most flexible and easy to use models for time series. The future value of a variable, in an AR model, is assumed to be a linear function of several past observations and random errors. Accordingly, AR models have proven to be especially useful for describing the dynamic behavior of economic and financial time series and for forecasting (Tsay, 2005; Zivot & Wang, 2006). Recent literature proved their superiority in financial modeling, in terms of accuracy, to most other techniques such as Neural Nets and SVM. (Adebiyi et al., 2014)

The basic p-lag vector autoregressive model (AR(p)) has the form:

$$\mathbf{Y_t} = \mathbf{c} + \mathbf{\Pi_1 Y_{t-1}} + \mathbf{\Pi_2 Y_{t-2}} + \cdots + \mathbf{\Pi_p Y_{t-p}} + \mathbf{\epsilon_t},$$
$$t = 1 \ldots T$$

with Π_i being the coefficients and ϵ_t is an unobservable zero mean noise. We can also do subset-autoregression where we pick multiple specific lags instead of a full range of lags like the basic AR(p) would do as show in the previous equation.

One of the most common extensions to Autoregressive models is to include moving averages and integration terms that take into account the stationarity of the time series. This is what constitutes Auto-Regressive Integrated Moving Average (ARIMA) models. They are the most general class of models for forecasting a time series which can be made to be stationary by differencing (if necessary), perhaps in conjunction with nonlinear transformations.

Accordingly, an ARIMA model is fully determined by 3 parameters, one for each of the (AR),(I) and (MA) components:

1. p = order of autocorrelation for (AR)
2. d = order of integration (differencing) for (I)
3. q = order of moving averages for (MA)

Determining these orders can be done either visually using the Autocorrelation Function plot (finds the autocorrelation coefficients mentioned earlier) and the Partial Autocorrelation Function plot (correlation conditioned on other lags' autocorrelation) [Figure 4.2.1] or analytically using a goodness of fit measure (which order fits the data best).

ACF or PACF plots indicates which lags are most correlated within the time series. They can be used to gauge if there is any sort of autocorrelation within the time series. They can be also be used to determine the exact orders depending on how the values in the plots decay or simply cut off but we're not covering those methods as we focused on the more methodical analytical approach.

For the goodness of fit, we had the options to use the Akaike information criterion (AIC), the Bayesian information criterion (BIC) or several other criteria that are more or less similar. These criteria would allow us to compare the goodness of fit of different models, indexed by their parameters. Accordingly, we would be able to decide the best order or parameters for our model based on the set that achieves the best fit. Later, we will discuss which subset of the data this fit was tested on.

For this paper, We decided to use the AIC:

$$AIC = -2 * \ln L + 2 * k$$

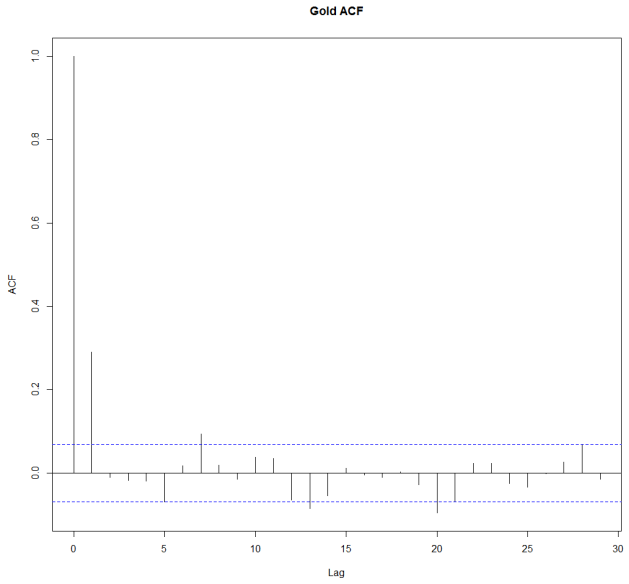

Figure 6. Autocorrelation Function plot for gold. Lags exceeding the dotted blue line have a statistically significant correlation with the 0-lag.Accordingly, the lags at 1, 14, 20 and 21 are correlated with the 0-lag (current time series).

where L is the maximum likelihood and k is the number of parameters. AIC aims to choose a model that minimizes the KL divergence between the true density and the density of the MLE of a fixed model.

Accordingly, the best model is the one with the lowest AIC. However, AIC is prone to overfitting and chooses predictive models over parsimonious ones. Given the literature's emphasis on the lack of predictive power in the domain of financial prediction, we decided to favor the predictive capacity over sparseness, for the order estimation, at least. BIC, on the other hand, favors sparsity over the predictive capabilities.

However, before testing the goodness of fit of different combinations of p and q parameters, we can figure out the d parameter using an Augmented Dickey Fuller test to verify the non-stationarity hypothesis of our data. Notice that random variable is stationary if its statistical properties are constant over time. Accordingly, it would have no trends and all of the variations around its mean would be random noise of constant amplitude. (Tsay, 2005; Zivot & Wang, 2006)

If our data is already stationary, then there is no need for differencing (determining by the d component). In our case, for example, the non-stationary hypothesis was rejected with a p-value of 0.01 in the ADF test. Therefore, the d parameter that best fits our data is 0.

4.2.2. Possible Extensions and Optimizations

One possible extension to ARIMA or any time series model, for that matter, is adding a Markov Switching Model that assumes the existence of latent regimes and builds an independent model for each regime because.(Hamilton, 1990) After testing, we couldn't find any significant regimes in our training data which might infer that our regimes are over longer periods of time than that spanned by our data.

A possible optimization, which we benefited from, is the use of simple linear models (or generalized linear models) to simulate the AR component of ARIMA in case the (MA) component was deemed superfluous. In this case, we can build a simple design matrix where each column is a lagged version of the response variable. This design matrix can be used for a standard Ordinary Least Squares regressions as well as various other models that are computationally faster than ARIMA (with respect to their R implementations, at least).

4.2.3. News Data: Exogenous Factors

Our news data that we extract from GDELT is considered an exogenous factor. In the time series literature, there have been several approaches to including such information into time series models. The most two prominent methods are impulse responses or simply considering the exogenous data as time series to be evaluated alongside the financial times series (the way you'd do in a regression) such as in ARIMAX models (ARIMA + Exogenous).

The impulse response method attempts to detect impulses, whose response or effect decays with time, within the time series data in order to localize the date of origin. Consequently, one would try to match the dates of such impulses with our news data in order to detect which columns in our data are most correlated with the big changes in those dates.

ARIMAX is simpler and more efficient, in literature and from our experience, especially with our linear models simplification. We add the columns of the news as extra time series whose autocorrelation with the response variable is evaluated in training. Note however that we use a lagged version of the news since our hypothesis is that the news from day D would affect the market after day D's closing which will be reflected on the difference of the values between D+1 and D.

4.2.4. Column Selection of News Data

However, a large number of columns in the news data (800 in the case of 100 clusters of GDELT data) can easily cause overfitting since the large number of variables could fit arbitrary data points to the noisy news columns. Accordingly, our out-of-sample testing results would suffer.

Another problem with a large number of columns is that ARIMA (the original implementation) and Ordinary Least Squares regression require the number of features to be less than or equal to the number of observations. This can be limiting as we would like to test different numbers of training days. Therefore, we recurred to Principal Component Analysis (among other methods that we tried) to reduce the dimensions. PCA converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.(Jolliffe, 2002) One can select a fixed number of principal components that express most of the variance in a certain data set. In our case, we selected a number of principal components that accounted for 99% of the variance in our data and used those principal components of the news data for the training as exogenous factors, instead of the original news columns.

4.2.5. L1-regularization: Lasso Regression

We also tried L1-regularization with Lasso regression since it inherently handles overfitting by enforcing sparsity, as compared to preemptively regularizing based on the variance, as in the case of PCA. (Friedman et al., 2009) Lasso is a least-squares regression with an L-1 penalty to

$$\hat{\beta}_{\text{lasso}} = \operatorname{argmin}_\beta \|Y - X\beta\|_2^2$$
$$\|\beta\|_1 < t$$

Figure 7. Lasso Regression

minimize the L-1 distance of the regression coefficients. Accordingly, Lasso is sparse since many elements of the array of coefficients is 0. Therefore, it eliminates the least important regressors. Accordingly, Lasso's sparsity is based on the actual importance of each regressor or news column with respect the response variable. The implementation (GLMNET), consequently, doesn't require the number of variables to be less than the observations which allows us to test a small number of observations without applying any dimension reduction beforehand. (Friedman et al., 2009)

Additionally, Lasso is computationally efficient as compared to other sparse regressions.(Friedman et al., 2009) Finally, we can select the degree of our sparsity by selecting a Lambda coefficient which helps us better gauge the change of the predictive power of our model.

Accordingly, we build Lasso model with the minimum Lambda as defined by cross-validation and then use that model to estimate our log-returns for each training window.

4.2.6. Testing Measures

For out-of-sample testing, we used two measures to evaluate the predictive capacity of the model and if it suffers from bias caused by the training data. The first is Mean Absolute Error which describes the difference in absolute value between our predictions and actual values. The closer MAE is to 0, the better we fit the data quantitatively. However, for our final purpose of designing a trading strategy, it could be good enough to detect if the stock is going up or down. In this case, predicting the binary values is what we seek. Accordingly, we would convert our predictions to binary based on their sign (positive log returns implies an increase from last day) and use simple binary-classification measures such as the accuracy rate. We focused on the accuracy rate because we have almost equal amount of positive and negative data points that we are not biased by the training which is usually the reason for evaluating recall, precision and F-1 measures.

4.2.7. Training the Time Series

The time series we had most success with is the spot price of Silver, in US Dollars between the dates of January 1st, 2013 and June 1st, 2015. We also converted our price values into log-returns as a way to normalize the values such as we model and forecast the log of the change rather than the values themselves. This is a common practice in financial modeling since it makes computations more efficient and comparison to other literature easier (scale independent). (Tsay, 2005)

$$LogReturn = \log \frac{X_t - X_{t-1}}{X_{t-1}}$$

We split our data into training and testing windows of varying sizes as can be seen in the results section. The windows guarantee the temporal order of the data points which is important for a time series analysis. We also resorted to a moving window technique, also called walk-forward optimization, which is a form of k-fold cross-validation for time series data. In this technique, we would slide the window on the training set of the time series data and re-estimate the coefficients at each iteration then look ahead by 1-step instead of multiple steps. Note that we're not re-estimating the order as we opted for estimating the order only once for the whole training dataset. This decision was motivated by the assumption that the order would be more or less constant over our small-to-medium range of dates and that re-computing it would be expensive and problematic to any future online implementations of this model or system.
Note that moving-window training is expected to have a higher predictive power than a simple fixed window technique since the model changes as the window moves and thus holds more significance as the predictions go further in time. With a fixed window, the further the predictions are from the window, the less relevant is the model from the fixed window since it's based on data that is too old.
Finally, we then collect the mean absolute error and the binary accuracy over all the iterations (different windows) and average them to evaluate the model for a given hyper-parameter. In our case, the main hyper-parameters was the size of the training window.

5. Results

5.1. Clustering

We manually inspect our clusters for both high similarity of articles within clusters and low similarity of articles between clusters. Since we do not have a gold standard for news event labels, we cannot calculate values such as true positives and true negatives. We present some qualitative results.

First ten articles of Cluster 0:

1. Retrial of 3 Al-Jazeera journalists (Egypt)
2. Protests for seizing of hospital accounts (DR Congo)
3. Waterloo homicide arrest (England)
4. Taliban vs Afghan army (Afghanistan)
5. $20 million funding for Wilmington pharmacy (Delaware)
6. Tiger farms violate Endangered Species Law (China)
7. DeSoto corruption trial (Mississippi)
8. Six die at David Owuor's Nakuru crusade (Kenya)
9. Temp workers fight for wages (Chicago)
10. Grant of bail to Lakhvi, mastermind of Mumbai attack (Pakistan)

As can be seen, purity for cluster 0 is fairly high, with seven of ten articles dealing with judicial decisions. Formally, purity is calculated by taking the most frequent class of each cluster, and then measuring accuracy by counting the number of correctly assigned points and dividing by N. In general, most clusters appear to have reasonable purity, but there is also high similarity between clusters. For example, articles similar to the first article in cluster 0 also appear in cluster 5. Cluster 5 is similar to cluster 0 in that they both deal with judicial decisions.

A search for the name of the mastermind of the Mumbai attacks, Zakiur Rehman Lakhvi, returned results in 69 of 1000 clusters, which indicates we have much more cross-cluster similarity than we wouldI like. But this is a more

preferable problem to have than a lack of intra-cluster similarity. Our eyeball test can only evaluate cluster quality based on keywords such as actor names and locations and general topics. Its possible that Lakhvi can appear in 69 clusters because the other metrics articles are clustered on, such as tone or the importance features, which an eyeball test cannot detect, do in fact separate Lakhvi related articles into that many clusters.

5.2. Regression

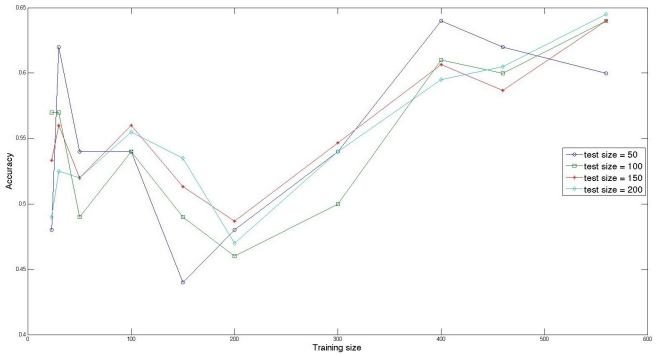

Figure 8. Accuracy for various test sizes

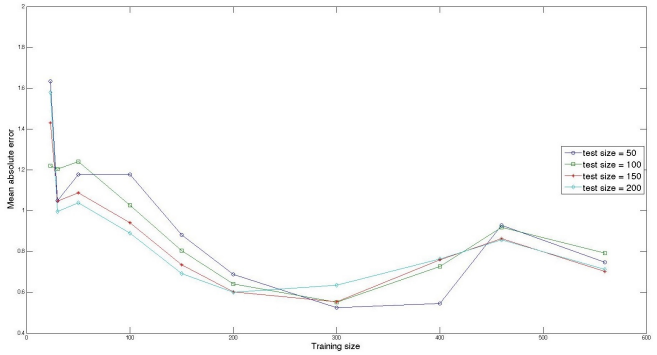

Figure 9. Mean aboslute error for various test sizes

Figure6 and Figure7 show the accuracy and mean absolute error across test size for a fixed window of last 22-560 training days. When we predict further in the future, our training data becomes less relevant and accuracy decreases, thus we also used a moving window of training set as shown in the two figures below.

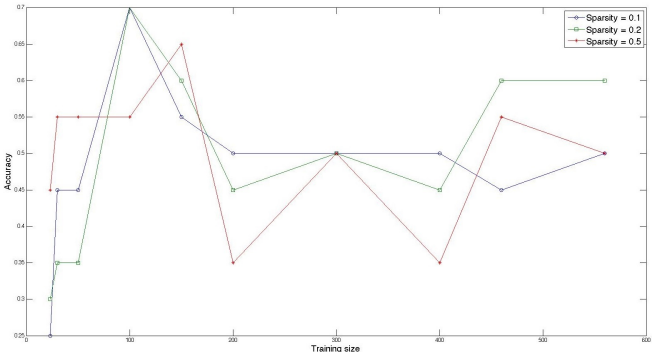

Figure 10. Accuracy for various sparsity

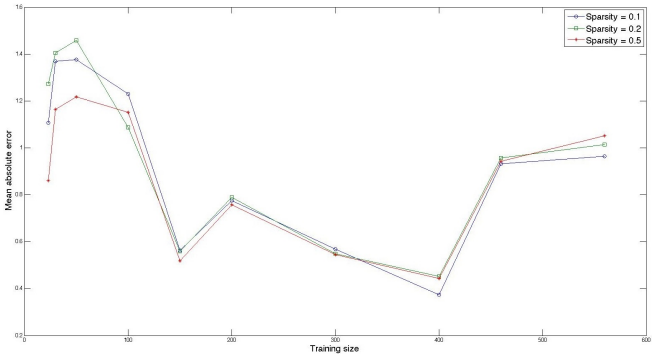

Figure 11. Mean abolsute error for various sparsity

Figure8 and Figure9 show the accuracy and mean absolute error across sparsity for a moving window of last 22-560 training days. As recomputing the whole windows is computationally expensive, we only recomputed the coefficients for the training days, but as we can see the best accuracies are approximately similar but we managed to reduce best mean absolute error ot 0.37 from 0.52 in the fixed window.

Figure10 is a log return plot with a moving window of 400 days and sparsity 0.1 having a mean absolute error of 0.37. We can see that qualitatively we can predict the positions of peaks and valleys but not the actual values.

6. Evaluation

To evaluate our results in the context of financial predictions, we ran a trading strategy on our test data set, using the predictions of our linear model.

Given a risk-free rate for borrowing leverage, r, the Kelly Criterion states that for a security with mean returns μ and volatility σ, the optimal fractional investment in the security one should make is $\frac{\mu - r}{\sigma^2}$ (Davis & Lleo). Estimating

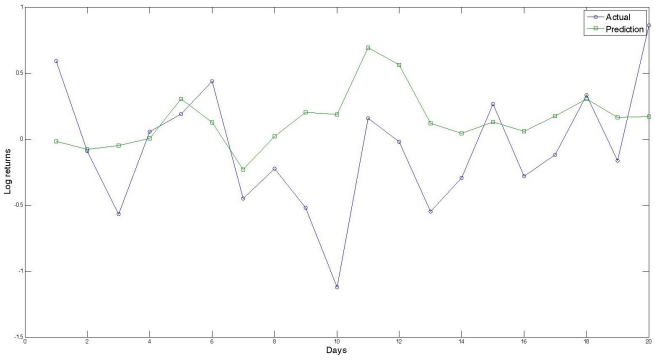

Figure 12. Comparison of log returns for sparsity = 0.1, training size = 400

volatility from the training set and using the predicted next-day returns to calculate the proportion of the commodity tested, silver, to buy, we were able to simulate our performance over the 221-day period:

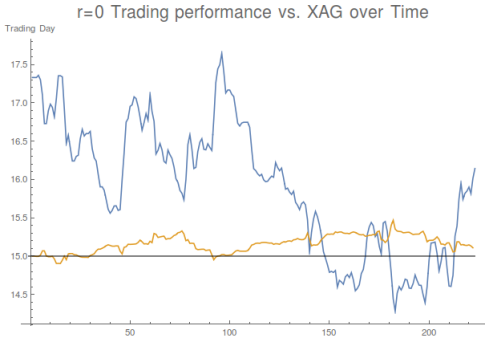

Figure 13. Portfolio performance for $r = 0$ over time on the test set. Note mistakes (when the net value of our assets, in purple, falls) typically occur from poorly predicting upswings. At the above rate, we make 0.7% returns on the given investment. We choose an artificial starting investment of $15 (allowing for partial stock purchases) to keep the scales the same. The black line is the no-risk play.

We don't use the S&P as a metric for the risk-free rate over this time because it actually fell 1.2% over the testing period. Using instead a generous estimate of the yearly U.S. Treasury bond yield for the year of the test set, at 0.5% (but modified to be compounded daily to fit with the per-day trading strategy), we still have some improvement.

The risk free rate did not affect our returns and is only useful as a benchmark to compare our strategy. Over the entire test run, we are only ever confident enough to purchase (or short) an amount of stocks equal to 51.8% of our current net value, so no leverage is ever used.

Because our model beats the risk-free rates, there is some (albeit small, financially speaking) predictive power offered by the model. The returns on our net worth fluctuate over the test period with a standard deviation of 0.740%, and we ended up with 0.727% gains at the end of the test period. This demonstrates performance from just the autocorrelations that can be inferred in the time series.

For comparison, when we run the ARIMAX model on the same time series with external GDELT information fed in, we lose 0.657% of the investment, with a standard deviation of the net worth being 0.27% of the original value. This outperforms S&P's loss of 1.2% (positive market alpha), but overall detracts from predictions based on autocorrelation alone.

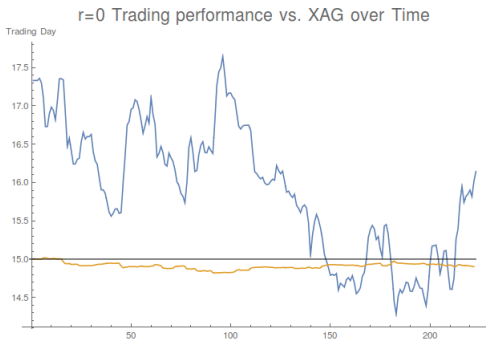

Figure 14. Portfolio performance for $r = 0$ over time on the test set. This is incorporating external GDELT information. The black line is the no-risk play (which at the time period makes money compared to investing in the market alone).

The small gains relative to the market are due to a large uncertainty in the estimates introduced by exogenous factors in the ARIMAX model, which are caused by lots of noise in the GDELT input data. These detract from accurate predictions based on autocorrelation alone. Overall, performance is hampered mainly by the large volume of irrelevant news. This contradicts the initial thesis which postulated that aggregate news data, at least when clustered by K-means, does not significantly inform next-day commodity pricing.

7. Future Work

7.1. Improving the Prediction Models

We have tried different window sizes for both training and testing as well as moving the windows instead of fixing them. We also tried using PCA for LM and ARIMA but found that using Lasso once the orders of AR and MA are determined to be more efficient. Our binary accuracy results are on-par with the state-of-the-art ones. However, it is clear that our prediction, for the simple log return data, is most from the time series lags itself and not GDELT. The

sparser our usage of GDELT is, the higher our testing results are. Therefore, we have to either analyze different time series (discussed later in this section) or try to estimate different features of the time series such as extreme values or peaks in the log return. For example, we could work on a logistic regression that classifies days in the top 10% in terms of log return value, as "high", days in the bottom 10% as low and the rest as "medium". In this case, we would be doing two classifiers, one on the "high" and one on the "low" in a similar fashion to exemplar SVM, where we train the high/low with so many negative examples that it has a higher certainty where predicting a peak. Our final result would indicate whether there are peaks (high or low) or if the trading activity is rather normal (medium) or if our results conflict (we get positive for both high and low). Only in the first case where we are sure about one kind of the peaks would we consider the prediction for our trading strategy.

7.2. Infinite Gaussian Mixture Model

For clustering news events we have relatively little information for deciding how many clusters there should be. In our K-means model, we do a parameter search for values of K from 10 to 5000 on a logarithmic scale. Ideally we would be able to use an infinite Gaussian mixture model that takes in a hyperparameter for a clustering coefficient and automatically determines the number of clusters and therefore remove the need for this imprecise parameter search.

We attempted using a Dirichlet Process GMM but the implementation we attempted to use was intractable given our computing power. We may attempt to optimize the parameters of the DPGMM in the future or work on limiting the model's training set further.

7.3. Filtering GDELT by Commodity

Up to now, we have been sampling the entire GDELT dataset for news events in order to predict the movement in commodity prices (e.g. silver). Since GDELT contains a large variety of news topics, the majority of the news articles within each day are not related to the specific commodity that we study. A more accurate way to predict the commodity prices would be to implement a filtering pipeline that removes irrelevant news items. This can be achived with natural language processing by selecting news articles that have keywords such as "silver" and "commodity". This approach should improve our prediction accuracy because it would remove a significant portion of the noise in GDELT.

8. Software and Data

All code used is available in an open-source repository.[1] GDELT provides free access to its database as well.[2]

Commodities pricing data is retrieved using the `R` package `quantmod`[3], which pulls historical prices using Yahoo.

References

Adebiyi, Ayodele Ariyo, Adewumi, Aderemi Oluyinka, and Ayo, Charles Korede. Comparison of arima and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics*, 2014, 2014.

Bollen, Johan, Mao, Huina, and Zeng, Xiaojun. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

Chen, Yu-Chin, Rogoff, Kenneth, and Rossi, Barbara. Can exchange rates forecast commodity prices? Technical report, National Bureau of Economic Research, 2008.

Davis, Mark and Lleo, Sébastien. Fractional kelly strategies in continuous time: Recent developments.

Friedman, Jerome, Hastie, Trevor, and Tibshirani, Rob. glmnet: Lasso and elastic-net regularized generalized linear models. *R package version*, 1, 2009.

Gidófalvi, Győző and Elkan, Charles. Using news articles to predict stock price movements. *Department of Computer Science and Engineering, University of California, San Diego*, 2001.

Hagenau, Michael, Liebmann, Michael, Hedwig, Markus, and Neumann, Dirk. Automated news reading: Stock price prediction based on financial news using context-specific features. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pp. 1040–1049. IEEE, 2012.

Hamilton, James D. Analysis of time series subject to changes in regime. *Journal of econometrics*, 45(1):39–70, 1990.

Jolliffe, Ian. *Principal component analysis*. Wiley Online Library, 2002.

Kamruzzaman, Joarder, Sarker, Ruhul, Ahmad, Iftekhar, et al. Svm based models for predicting foreign currency exchange rates. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pp. 557–560. IEEE, 2003.

[1] `https://github.com/vlad17/COS513-Finance`

[2] `http://data.gdeltproject.org/events/index.html`

[3] `http://www.quantmod.com/`

McQueen, Grant and Roley, V Vance. Stock prices, news, and business conditions. *Review of financial studies*, 6 (3):683–707, 1993.

Phua, Clifton, Feng, Yuzhang, Ji, Junyao, and Soh, Timothy. Visual and predictive analytics on singapore news: Experiments on gdelt, wikipedia, andˆ sti. *arXiv preprint arXiv:1404.1996*, 2014.

Schumaker, Robert P and Chen, Hsinchun. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):12, 2009.

Simon, Eric. Forecasting foreign exchange rates with neural networks. Technical report, Swiss Federal Institute of Technology, 2002.

The GDELT Project. Gdelt. `http://gdeltproject.org/data.html`, a. Accessed: 2015-10-28.

The GDELT Project. Gdelt 2.0: Our global world in realtime. `http://blog.gdeltproject.org/gdelt-2-0-our-global-world-in-realtime/`, b. Accessed: 2015-12-1.

Tsay, Ruey S. *Analysis of financial time series*, volume 543. John Wiley & Sons, 2005.

Zivot, Eric and Wang, Jiahui. Vector autoregressive models for multivariate time series. *Modeling Financial Time Series with S-PLUS®*, pp. 385–429, 2006.