AVA - Advanced Volatility Arbitrage

Alex Fine and Guy Wuollet

June 2019

1 Abstract

AVA is an AI powered volatility arbitrage infrastructure designed to stabilize high risk assets. The technology is currently applied to cryptocurrencies, however it can be applied more broadly to any volatile asset class. The output of AVA is a single number which indicates with high accuracy where the market is expected to move over the next n minutes. Beneath this output lies a highly modular & dynamic technical framework which allows for real-time predictions with a data latency of less than 15 seconds.

Given the modularity of the AVA system it can be used for a variety of tasks beyond the initially intended volatility arbitrage use-case. The system can be used to research purposes on the theoretical limits of predictive intelligence, serve as the core of a Reinforced Learning based trading strategy, and assist fundamental traders in entering markets at opportune times. The versatility of the AVA technology is discussed further in section 8.

2 Introduction

2.1 Problem

Today cryptocurrencies are a highly volatile asset class. If an investor is interested in entering these markets they effectively need to write off the cryto chunk of their portfolio and only view the enter & exit price on their investment. They cannot analyze the Sharpe on their investment because the σ is much too high. This reality is ludicrous. As of writing crytocurrencies have a combined market cap of $251 Billion, and the asset class holds a promising future. They have orthogonal returns to the S&P 500 and thus could significantly reduce aggregate portfolio risk if accessible as a usable asset. The fact that the majority of institutional investors can't touch crypto implies that massive market inefficiencies exist.

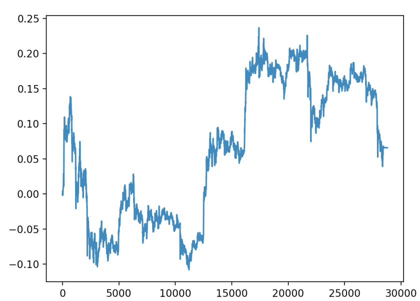
Furthermore, the traditional methodologies for asset stabilization do not work on cryptocurrencies due to the nuanced nature of the asset class. The most notable flawed yet popular idea to address is the idea of a crypto ETF, or an S&P 500 for crypto. This idea is flawed for two reasons. The first is that the beta among currencies is far to high to stabilize the asset class by bundling. The second reason this approach is poor is because if measured by market cap, the ETF would inherently contain "shitcoins." These are coins savvy investors should not hold, yet still dominate the charts.

Any viable solution must possess the capabilities to stabilize a singular asset, as opposed to simply leveraging the ramifications of the efficient market hypothesis (EMH) as applied to cryptocurrencies.

2.2 Solution

To stabilize these assets I built AVA. AVA's AI is designed to continuously monitor market signals & provide insights as to the direction of the market n minutes into the future. When the AI is very confident of an upcoming market move or a region of high volatility, it can move money into a stable base currency, long, or short the currency accordingly. The AI is trained on data from numerous currencies and thus applies translational learnings between them. This means that AVA can intelligently predict how a Bitcoin move three days ago will effect how Litecoin is going to move in the next 5 minutes.

In short, we turned the asset on the left, into the asset on the right.



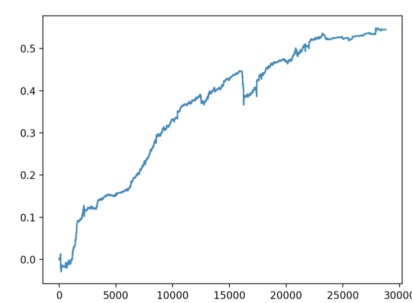


Figure 1: Litecoin chart. x axis = minutes from t_0, where t_0 is sometime on May 16th. y axis = total change as a fraction of original price

Figure 2: An equivalent investment with AVA. Axis the same.

AVA has the capabilities to stabilize highly volatile assets.

2.3 Investible Universe

AVA has the capability to trade any coin with enough liquidity. Today, this includes BTC, ETH, LTC, BCH, XRP, ETC & ZEC if traded from coinbase pro. The system has the capabilities to make predictions across, 1 minute, 5 minutes, 15 minutes, 30 minutes, 1 hour, 2 hours, 4 hours, 6 hours, 12 hours, and 24 hours. Thus, when I refer to n in this paper, what I'm really saying is $\forall n \in \{1, 5, 15, 30, 60, 120, 240, 360, 720, 1440\}$. Much of this paper focuses on models & algorithms related to trading LTC with predictions for 5 minutes in the future. I chose this combination because there is relatively high inefficiencies at this time scale, and LTC is a rather unstable coin. Thus, this combination can highlight the benefits of the AVA system quite well. That being said, LTC & 5 minutes are in no way unique to AVA. It can perform similarly well on ETH @ 2 hours.

2.4 Market Analysis

Crypto-markets are fundamentally different from traditional financial markets. Crypto-markets are incredibly volatile and tend to move in distinct spikes, gaining or losing value all at once before quickly plateauing. Crypto-markets are also frequently manipulated, leading traders to attribute these spikes to manipulation and vice versa. These characteristics create markets that are not only inefficient, but able to be arbitraged.

For the remainder of this analysis we will focus on Bitcoin specifically because it is the hardest cryptocurrency to manipulate, the safest, most liquid, and the easiest to find counter parties to trade with. Each analysis below was done using the same data that powers the neural network model used to trade.

2.4.1 Distribution of Returns

We will examine many flavors of return distributions, but begin with the most straight forward. The below charts show the distribution of Bitcoin returns for different time scales. `return_1` shows the distribution of returns from 1 minute ago, `return_10` shows the distribution of returns from 10 minute ago, and so on. Note that the distributions become more normal as the time scale increases. On small time scales, most returns are zero, but there are still a meaningful number of 10, 25, and 100 sigma events. As the time scales gets bigger, returns are not as tightly cluster towards 0, but there are fewer extreme events.

2.4.2 Distribution of Z-Score Returns

Next we will examine the distribution of Bitcoin return z-scores. This matters because most traditional financial models assume that returns are normally distributed. This is not true in traditional financial markets, as returns distributions are better modeled with a fat tail. This inaccuracy leads many to poorly attribute risk and make incorrect assumptions. We will again look at the distribution of returns(with z-score values this time), but also include a chart

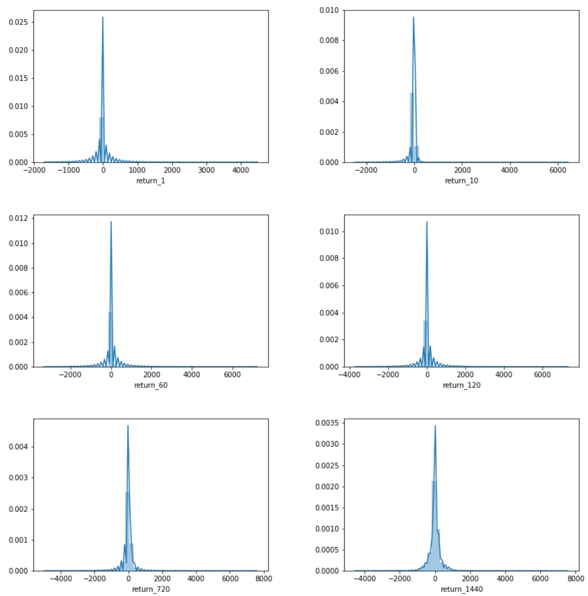


Figure 3: Bitcoin return distributions for different time scales. return_n is the return from n minutes ago.

showing how many n-sigma events there were for different values of n. Because there are so many possible return events, it is easier to see the distribution numerically when looking for outliers.

The z-score for any data point is calculated by that point minus the mean of the data over the standard deviation of the data. Mathematically, where μ is the mean of the data and σ is the standard deviation of the data, the z-score is

$$(x - \mu)/\sigma \tag{1}$$

Z-score represents the number of standard deviations above or below the mean a given data point is. A z-score above n is referred to as an n-sigma event in the below chart. The chart separates positive and negative n-sigma events in each cell.

The number of n-sigma events tends to increase as the time scale increases. This makes sense, as the market has more time to ingest news and change prices based on fundamentals. However, note the 50-sigma and 100-sigma columns. The number of 100-sigma events is highest when analyzing 10 minute returns. This indicates that some news is not baked into the market immediately, and takes more than 1 minute to fully affect the market. This same effect can be seen for 50-sigma events, where the most 50-sigma events occur with a 2 hour return timescale. In both cases, the number of 50-sigma and 100-sigma events drops to zero over a larger time scale. This indicates that there is alpha

Table 1: n-sigma events for Bitcoin Return Distributions

n-sigma	**1**	**2**	**3**	**5**	**10**	**50**	**100**
return_1	+29170 / -29627	+7185 / -7506	+2576 / -2815	+567 / -623	+117 / -125	+11 / -12	+3 / -1
return_10	+23156 / -23822	+5923 / -6577	+2308 / -2712	+771 / -802	+144 / -242	+15 / -9	+10 / -0
return_60	+24123 / -25595	+6226 / -7684	+2280 / -2804	+656 / -968	+241 / -378	+55 / -0	+0 / -0
return_120	+25665 / -29804	+5984 / -7786	+1917 / -2665	+618 / -1080	+194 / -247	+68 / -0	+0 / -0
return_720	+40352 / -47558	+10057 / -11838	+2549 / -4106	+798 / -1078	+695 / -238	+0 / -0	+0 / -0
return_14400	+47361 / -59974	+14723 / -15790	+2539 / -4167	+1440 / -1295	+700 / -68	+0 / -0	+0 / -0

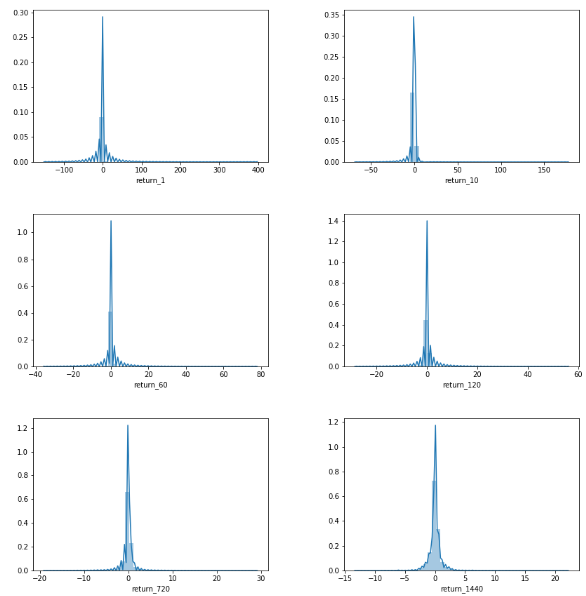


Figure 4: Z-score of Bitcoin return distribution for different time scales. return_n is the return from n minutes ago.

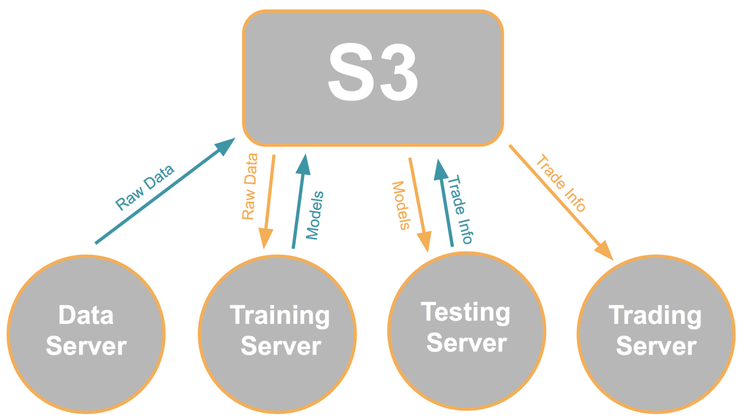
to be captured beyond high frequency, super low latency, trading strategies. It also indicates that there are huge events in the market. 50 or 100-sigma events should be generational outliers in a normal distribution return model. This data is generated from 15 months on Bitcoin price data, and reflects that crypto-markets have generational return events on a regular basis. Because these events take time to ripple out into the market, they should be predictable, as some signal takes time to be understood by the market before causing massive price movements.

2.5 Architecture

There are two key components of the AVA architecture. The first is the high level relationship between computational & informational servers. The second is the internal architecture of each computational server.

2.5.1 Relational Server Architecture

AVA is discretized into five servers, four computation servers & one information server. Each server has a highly specific purpose, and the computation servers have their own inputs and outputs (similar to a function). They also do not directly interact with each other - again behaving as clean independent functions as opposed to one glob of code. This design allows for modularity between components, and reliability in case one component breaks. The architecture can be visualized as follows:



The server flow starts with the data server. The data server pulls in live market data every three hours and preprocesses it as necessary. Then, it pushes all of that data to the S3. The training server starts by pulling down the necessary data from S3. It then takes this data and trains predictive models on the data. The training server then pushes just the best models to the server. The testing server downloads those models and analyzes them. After producing optimal investment criteria per a specific model, it uploads that information to S3. Finally, the trading server downloads the optimal investment information for the best model. It then trades on the market.

This paper's structure is inspired by the server architecture. From here forward, the paper linearly discusses four discrete sections on the data, training, testing, & trading.

3 NLU

In addition to price data, we decided to explore alternative data for cryptocurrencies. This could have taken many forms, but we chose to highlight natural language as a way to gain insight about crypto-market movements. Crypto-communities are often fervent in their belief and evangelistic in tendency. They are out-spoken and actively want to make their views heard. As such, it makes sense for their to be an inherent sentiment of any crypto-community, a sentiment that should directly affect price. This sentiment may be baked into the market price already. However, shifting sentiment can change slowly, without anyone consciously trading on the information. Like a frog boiling in hot water, active community participants may not notice a distinct difference from any previous day. Since crypto-markets tend to move with sudden spikes, in both the positive and negative directions, this indicates there is a tipping point. Identifying this tipping point could be a fantastic predictor of crypto-currency price.

After identifying crypto-community sentiment as a source of alternative data, we chose to use Reddit as the primary source. Other communities like Discord or Telegram are much harder to aggregate data for and communities like Twitter lack inherent structure. When deciding between Twitter and Reddit, Reddit has clear advantages. Reddit communities are based around specific topics and have an aggregate sentiment over time. They are also clearly aligned with different crypto projects and have clear opinions on topics. Selecting the right profiles to follow on Twitter is a nightmare. There may be important profiles missed, or useless profiles included. The relevant profiles may change over time, requiring any model to be re-trained over the new set of profiles. Taking only specific posts with given terms or hashtags also limits the model a priori. Finally, including personal profiles will train the model on tweets that are irrelevant to crypto-markets. Plenty on influential crypto figures tweet about their kids, sports, food, or other passions, all useless information from the perspective of the market. This could create spurious correlation and bias models. Reddit communities, called subreddits, solve all of these problems.

3.1 Data

We aggregated all comments ever posted on Reddit, from it's inception to the end of February 2019. For this, we used Pushshift, a subreddit dedicated to archiving Reddit's history and making it available through API. In total, this amounted to more than a terabyte of data, way too much to analyze without serious infrastructure. To reduce the projects scope, only data from crypto-currency subreddits was analyzed, with a specific focus on r/Bitcoin and r/btc. r/Bitcoin is a subreddit specifically focused on Bitcoin. r/btc is a subreddit that supports BitcoinCash. The subreddit believed in larger block sizes during the hard fork between Bitcoin and BitcoinCash and has since become a BitcoinCash focused community. These subreddits were chosen because of their focus on Bitcoin from different viewpoints and the potential for Bitcoin and BitcoinCash as a trading pair.

3.2 Methods

Initially, standard sentiment classifiers were tried on the data. These classifiers included TextBlob and different NLTK classifiers. However, they performed poorly. The classifiers were trained on a domain agnostic corpus of English text. Because words and phrases have different meanings in different contexts, the classifiers didn't understand most of the comments posted on r/Bitcoin and r/btc.

To train a domain specific classifier we used project called SocialSent: Domain Specific Sentiment Lexicons for Computational Social Science [1]. A domain specific classifier is a sentiment classifier that understands a given context and classifies text within that context. We choose to use SocialSent because it only required 20 million tokens of text (a token is a word, number, etc) and could be trained unlabeled data. The standard sentiment classifiers were trained with labeled data, a process that requires million of hand annotated pieces of text. This process is time consuming and hard.

SocialSent begins with a vector-space model for token co-occurrence. For this project, two tokens were considered to co-occur if they both occurred in a given comment. This result is a $n \times n$ matrix were n is the number of unique tokens. Because there were more than $50,000$ tokens and a matrix with $50,000^2$ tokens is too many, a vocabulary of the top 5000 most frequent tokens was chosen, resulting in a 5000×5000 matrix. Automatically included in this vocabulary were specific seed words that are later used to induce the polarity of token within the sentiment lexicon. That resulting matrix is then reweighted using two methods. First, PPMI is applied to it. PPMI (postivie pointwise mutual information) and intuitively rebalances the matrix based on how often a term should have co-occurred given the other co-occurences in the lexicon. Mathematically it is defined by

$$M_{i,j}^{PPMI} = max(\frac{p(w_i, w_j)}{p(w)p(w_j)}, 1) \tag{2}$$

where p is the empirical probability of term co-occurrence.

Next the singular value decomposition of that matrix is found.

$$M^{PPMI} = U \sum V^T \tag{3}$$

And the embedding for each term w_i is

$$w_i^{SVD} = (U)_i \tag{4}$$

These embeddings have dimension 300, as is standard practice for the method. The embedding space is then translated into weighted graph where a term is connected to its k nearest neighbors with edge weights

$$E_{i,j} = arccos(-\frac{w_i^T w_j}{\|w_i\|\|w_j\|}) \tag{5}$$

Sentiment polarity is then populated across this graph using a random walk method from different seed words. For this project, the same seed words SocialSent used in their Twitter and Reddit models were used. The positive seed words were love, loved, loves, awesome, awesome, nice, amazing, best, fantastic, correct, happy. The negative seeds words were hate, hated, hates, terrible, nasty, awful, worst, horrible, wrong, sad. The positive and negative polarities from the positive and negative random walks are combined into one score such that p^P is the positive score and p^N is the negative score

$$\frac{p^P(w_i)}{p^P(w_i) + p^N(w_i)} \tag{6}$$

As a result, each of the 5000 most frequent tokens have a polarity induced upon them.

To classify the sentiment of a new comment, it was tokenized in the same fashion as the text used for training (using NLTK TweetTokenizer) and then polarity was applied to each word. The polarity of a comment was the average polarity of all sentiment bearing tokens in the comment. For example, "I love Bitcoin" would yield a positive sentiment because the polarity of I is neural, the polarity of love is positive and the polarity of Bitcoin is neutral.

3.3 Preliminary Results

The sentiment of each comment was then transformed using a moving weekly z-score and performed on the price of Bitcoin. The below graphs show the price of Bitcoin in the hour immediately following a 25-sigma Reddit sentiment event. It demonstrates a result that is consistent across time scales, sentiment classified with the domain specific polarity lexicon is indicative about 15% of the time.

3.4 Future Work & Lessons Learned

While, there is some signal resulting from the custom trained, domain specific, polarity lexicon, the signal is not as strong as expected. Simpler methods could possibly achieve better results. During data exploration, volume of total posts and volume of specific terms correlated highly with price and volatility. These indicators are easier to find, less of a black box, and could yield better results. Entity recognition could also be significantly useful, helping to distinguish between the different flavors of Bitcoin. For example, is Bitcoin Bitcoin, BitcoinCash, Litecoin, ZCash, or Bitcoin Satoshi's Vision? How does the mention of one token affect the price of others? These more basic questions seem more likely to yield explainable results that could deliver returns.

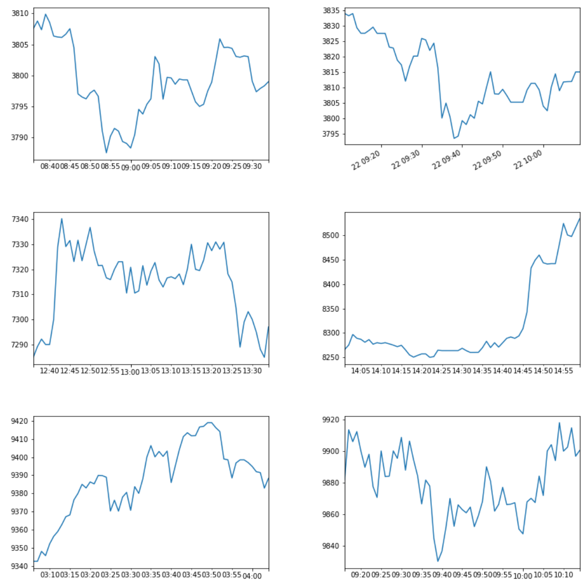


Figure 5: Bitcoin Price 1 hour immediately after 25-sigma sentiment event using weekly moving z-score

4 Data

4.1 Overview

The AVA system relies on per minute price data from BTC, ETH, LTC, & BCH. It preprocesses this raw data before use in the RNN.

4.1.1 Raw Data

The data server reads in raw data from the “cryptocompy” python API wrapper. Only the most recent day of per minute data is available from an API call, and the internal data pipeline must stay up-to-date. Therefore, the data server calls this API & stores the results every three hours. A separate error server monitors that this server is properly online. If the server goes down I get a text.

Getting the most recent price data is as simple as:

```
price.get_historical_data(currency, 'USD', 'minute', aggregate=1, limit=(500))
```

4.1.2 Pre-processing

After the raw data has been recorded, the AVA system produces price change and volatility derivative metrics to use as inputs into the RNN. The function to generate an input, X_{ij} solely depends on the price array. It can be calculated as follows:

$$X_{ij}(\boldsymbol{p}) = \Big[\Big(\begin{bmatrix} p_i \\ p_{i-j} \\ p_{i-2j} \\ \dots \\ p_0 \end{bmatrix} \cdot \big[(\tfrac{11}{13})^0 \quad (\tfrac{11}{13})^1 \quad (\tfrac{11}{13})^2 \quad \dots \quad (\tfrac{11}{13})^i\big] \cdot \frac{2}{13}\Big)^{-1} \cdot p_i - \overline{\mu}(X_{ij})\Big] \cdot \frac{1}{\overline{\sigma}(X_{ij})}$$

To put simply, an individual exponentially moving average (ema) derivative data-point for time i is generated by taking the price at time i and dividing it by the ema, and then normalizing by the expected value and standard deviation of that random variable. The final two steps exist in order to have all distributions centered at zero with std. of 1. This calculation must be performed repeatedly to generate each individual i, j combination of data. As of writing, there are 33.6 million of these combinations.

Volatility metrics are calculated as a derivative on the ema metrics. It requires four inputs - the per minute high, low, price, and Xi value. The array of volatility data is calculated as follows:

$$\boldsymbol{V}(\boldsymbol{X_j}, \boldsymbol{h}, \boldsymbol{l}, \boldsymbol{p}) = \sqrt{\frac{|\boldsymbol{X_j} - \boldsymbol{l}| \cdot |\boldsymbol{X_j} - \boldsymbol{h}|}{\boldsymbol{p}^2}}$$

4.1.3 Data Quantity

The most recent sizes for the data matrices are as follows:
Raw: 692397 x 3 $= 2,077,191$ net pieces of raw data
RNN Input Data: 692397 x 88 $= 60,930,936$ net pieces of input data

5 Training

5.1 Overview

AVA uses an RNN-LSTM to make predictions as to where the market is going to move n minutes in the future. RNN-LSTM's are notoriously good at modeling time series data, such as asset price data, due to their ability to discount old data while factoring in previous market moves. AVA tests on the most recent fifteen days of data, and trains on everything else.

5.2 Model

5.2.1 RNN-LSTM's Unique Effectiveness on Crypto Data

If a series of random variables (such as 5 minute crypto returns) does not limit to a normal distribution, that implies that the series of random variables is not independent and identically distributed, but instead the variables are somehow dependent upon each other. This dependence is especially true for crypto given much higher rates of volatility clustering than in traditional stock markets. See the data analysis section for more on this effect.

An RNN-LSTM model nulls the Markov assumption which many NN make. That is it does not assume $P(X_i = x|X_{i-1}, X_{i-2}, ..., X_0) = P(X_i = x|X_{i-1})$. This has the effect that RNN-LSTM's work very well on data where $P(X_{100} = x)$ might depend on X_{89}, X_{74}, X_2. It works very well on highly dependent series of random variables. Which, as we showed in the data section, perfectly describes the behavior of cryptocurrency returns as a RV.

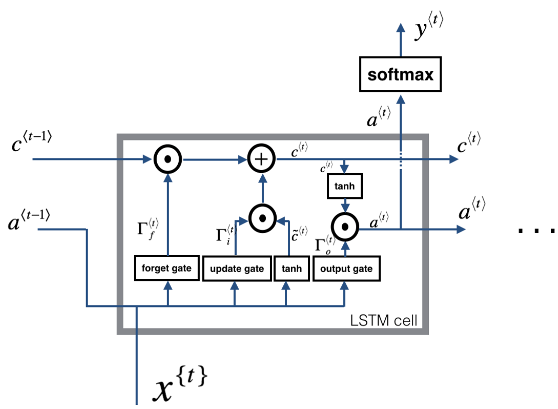
5.2.2 Underlying Math

AVA uses a traditional RNN-LSTM model. Many of novelties of AVA are not in the uniqueness of the model design itself, but rather in the data used, the tuning measures employed, and the comprehensive analysis afterwards.

What is unique about RNN's in comparison to other NN's is their ability to factor previous data into the prediction. It achieves this through the use of an additional weights matrices - illustrated below:

$$h_t = \sigma(W^{(hh)}h_{t-1} + W^{(hx)}x_t)$$
$$\hat{y}_t = Softmax(W^S h_t)$$

The LSTM layer discounts old data in order to weight newer market data higher. An illustration of this process is as follows:



$$\Gamma_f^{\langle t\rangle} = \sigma(W_f[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_f)$$
$$\Gamma_u^{\langle t\rangle} = \sigma(W_u[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_u)$$
$$\tilde{c}^{\langle t\rangle} = \tanh(W_C[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_C)$$
$$c^{\langle t\rangle} = \Gamma_f^{\langle t\rangle} \circ c^{\langle t-1\rangle} + \Gamma_u^{\langle t\rangle} \circ \tilde{c}^{\langle t\rangle}$$
$$\Gamma_o^{\langle t\rangle} = \sigma(W_o[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_o)$$
$$a^{\langle t\rangle} = \Gamma_o^{\langle t\rangle} \circ \tanh(c^{\langle t\rangle})$$

5.3 Optimization Methodology

To optimize the hyperparameters of the RNN we employed the technologies of SigOpt. Doing so illuminated the sensitivity of the model to hyperparameter tuning. Without enough time to train a certain model, or with an incorrect learning rate, the model couldn't beat random guessing. However, with properly tuned parameters it could predict quite well. There were three naturally occuring phases to optimizing with SigOpt.

5.3.1 Phase 1

During first training round we did not give the models enough time to tune. This resulted in generally convoluted and often confusing results. However, the parameter importance data corroborated our expectations thus indicating the system was functioning correctly.

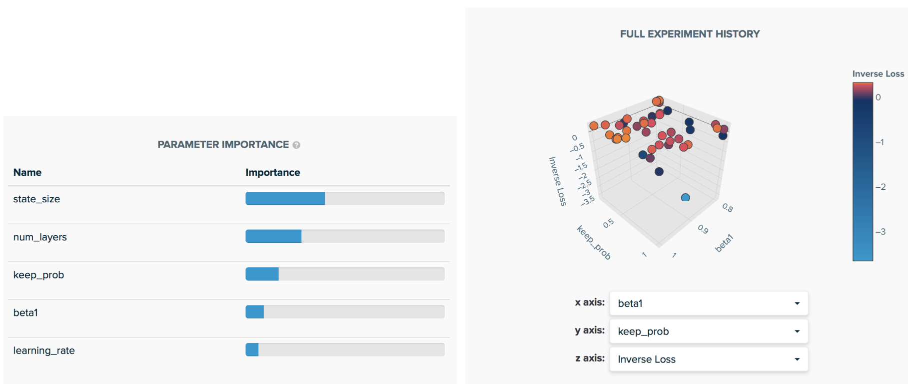


Figure 6: The relative importance of each hyperparameter on the left, the the distribution of results on the right

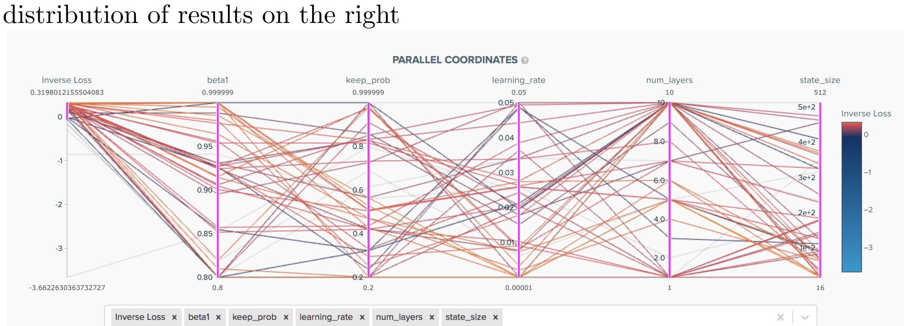
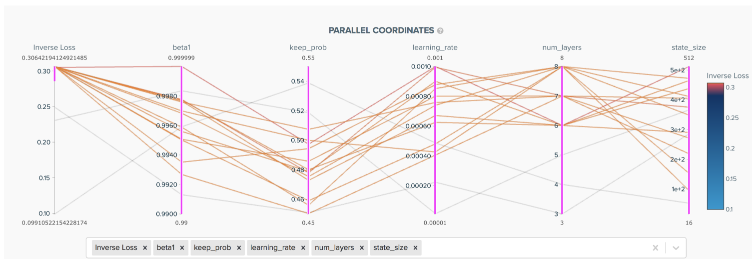


Figure 7: Results remained unclear

5.3.2 Phase 2

After giving the models enough time to train we were able to discern clear patterns in how to best optimize some of our hyperparameters. However, we still didn't know how to best optimize the number of layers or state size.



5.3.3 Phase 3

On our final batch of training, where we gave each model considerably more time to tune, we were able to identify very clear patterns in how to best optimize the models.

5.4 Results

This paper outlines results from two combinations of currency and investible periods, BTC 120 and LTC 5.

5.4.1 BTC 120 Minute

State Size	**Num Layers**	**Test Accuracy**	**Test Alpha Accuracy**
125	5	56.79%	49.78%
286	3	52.27%	50.51%
366	5	52.91%	50.83%
52	4	54.15%	51.45%
83	1	57.87%	50.32%

5.4.2 LTC 5 Minute

Note that the model represented on the fourth row had the same hyperparameters as the model on the fifth row, the only difference is that the fifth row model had significantly more time to train.

State Size	**Num Layers**	**Test Accuracy**	**Test Alpha Accuracy**
256	2	53.32%	51.51%
299	1	53.79%	51.74%
321	2	53.41%	51.55%
388	1	53.90%	51.80%
388	1	62.77%	54.59%

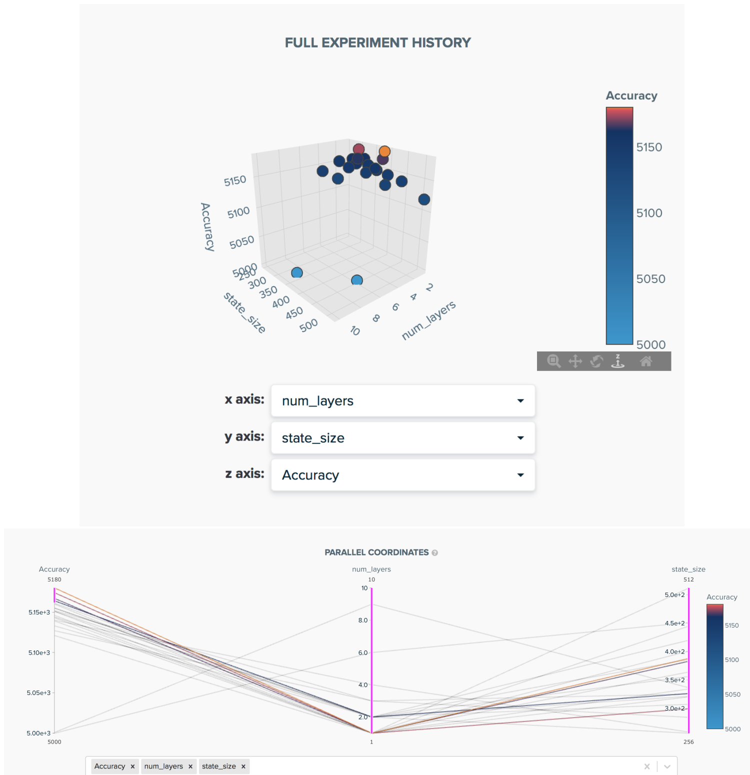


Figure 8: Superior Models in Orange

6 Testing

6.1 Approach

During the testing phase the goal is to understand how to best use the trained model to make predictions & optimize the Sharpe on trading. To do so I built multiple functions to optimize the buy/sell prediction point, and others to identify the highest accuracy prediction points. However, these numbers alone paint a clouded picture of what's really going on. To understand the testing phase it is first critical to understand the data we're analyzing, and how to use that data to inform intelligent predictions.

The output data which is analyzed in testing is an array of length 28800x3. The first column is how the market behaved over a 5 minute interval, and the second two columns are how we predicted the market would behave. Our predictions are in the range (0,1), which represents AVA's confidence that the market is going to go up or down. A sample subset of data is below:

Market % change	Negative Prediction	Positive Prediction (1 - negative)
0.0967305088024708	0.44528747	0.55471253
0.22228665313617857	0.40866876	0.5913313
0.13534416086620318	0.52520645	0.4747936
0.2896591677126554	0.3851425	0.6148575
0.1641718976339949	0.44165558	0.55834436

To effectively trade we need to know which subsets of our predictions are most valuable, and by how much. To discover this subset we're going to analyze recall over different confidence intervals.

6.2 Market Subsets

In this section we analyze subsets of our total prediction set. Subsets are determined by the prediction confidence.

6.2.1 Full Sample Space Ω

Over the complete sample space:
Positive Recall: 62.12%
Negative Recall: 63.32%

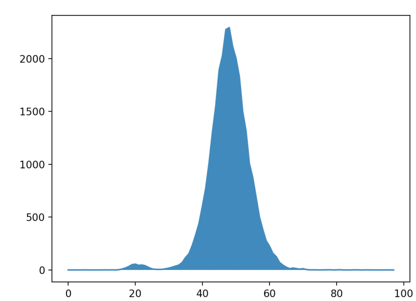


Figure 9: PDF of All Predictions (predictions scaled up by a factor of 100)

6.2.2 $\{X_i | X_i \in \Omega \wedge (X_i < 0.45 \vee X_i > 0.55)\}$

Only analyzing predictions below 0.45 or above 0.55:
Positive Recall: 74.81%
Negative Recall: 70.01%

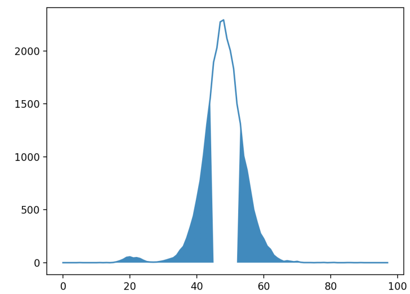


Figure 10: Subset of predictions at 0.55 or above in confidence

6.2.3 $\{X_i | X_i \in \Omega \land (X_i < 0.40 \lor X_i > 0.60)\}$

Only analyzing predictions below 0.40 or above 0.60:
Positive Recall: 80.81%
Negative Recall: 78.23%

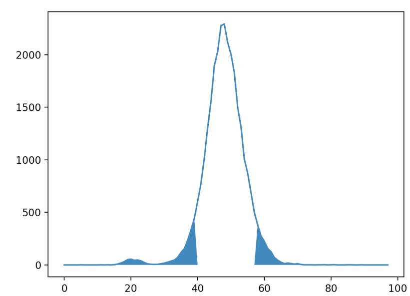


Figure 11: Subset of predictions at 0.60 or above in confidence

6.2.4 $\{X_i | X_i \in \Omega \land (X_i < 0.35 \lor X_i > 0.65)\}$

Only analyzing predictions below 0.35 or above 0.65:
Positive Recall: 78.14%
Negative Recall: 89.66%

Notice the positive recall dropped a little. I theorize this drop can be attributed to a smaller sample size and thus more variance in performance.

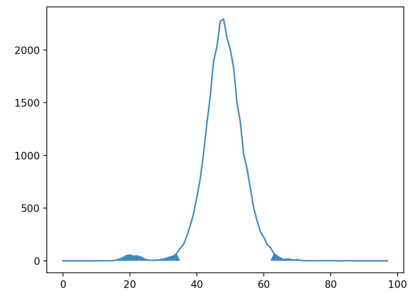


Figure 12: Subset of predictions at 0.65 or above in confidence

6.2.5 Conclusion

We're going to analyze the model performance on two sets, Ω and all predictions of greater than 0.6 confidence. A choice of 0.6 allows for enough of a sample to get statistically significant returns, while maintaining discretion on which trades get executed. Analyzing results at 0.5 allows us to compare the effects of limiting trade to a subset of predictions vs. always trading.

6.3 Fee Based Analysis

In this section we analyze how the model would have performed in the market during the extended test set (because data updating is live we could analyze more than just the test set of unforseen data). We specifically analyze the performance given different fee structures.

6.3.1 Current Fee Structure - 25 bips loss/trade

If AVA traded today without any intelligent market entry algorithms it would face the common exchange fee of 25 bips per trade.

Returns & Sharpe with a fee of 2.5¢ per trade on every $10 invested:

	0.5 Thresh	**0.6 Thresh**
Returns:	-488.42%	-58.99%
Sharpe:	-19.9	-24.28

6.3.2 $10k Traded - 10 bips loss/trade

If AVA ran with $10k, again without any intelligent market entry algorithms, it would face an exchange fee of 10 bips per trade.

Returns & Sharpe with a fee of 1.0¢ per trade on every $10 invested:

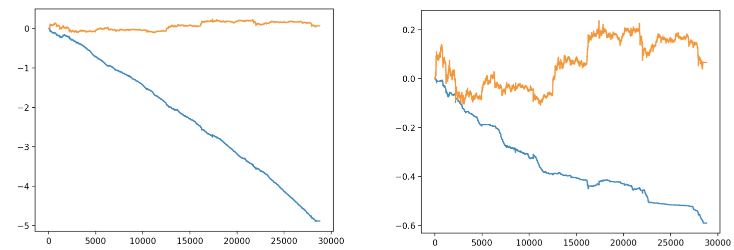


Figure 13: 0.5 Left, 0.6 Right

	0.5 Thresh	**0.6 Thresh**
Returns:	-111.82%	-4.50%
Sharpe:	-5.05	-2.04

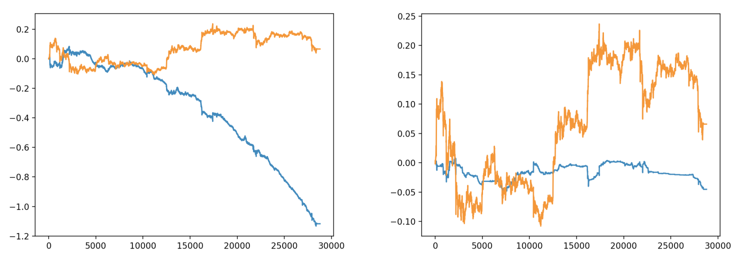


Figure 14: 0.5 Left, 0.6 Right

6.3.3 $1MM Traded - 5 bips loss/trade

If AVA ran with $1M, again without any intelligent market entry algorithms, it would face an exchange fee of 05 bips per trade.

Returns & Sharpe with a fee of 0.5¢ per trade on every $10 invested:

	0.5 Thresh	**0.6 Thresh**
Returns:	13.89%	13.65%
Sharpe:	0.639	6.27

6.3.4 Highly Efficient Entry - 4 bips loss/trade

If AVA ran with intelligent market entry algorithms today it would face an exchange fee of < 4 bips per trade. 4 bips per trade is the lowest cost achievable

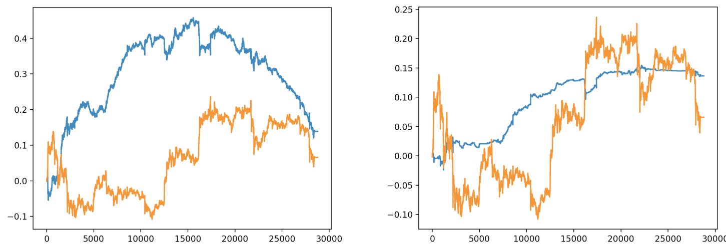


Figure 15: 0.5 Left, 0.6 Right

by exchangify, who I've partnered with to route all my trades through.

Returns & Sharpe with a fee of 0.4¢ per trade on every $10 invested:

	0.5 Thresh	**0.6 Thresh**
Returns:	39.01%	17.27%
Sharpe:	1.79	7.95

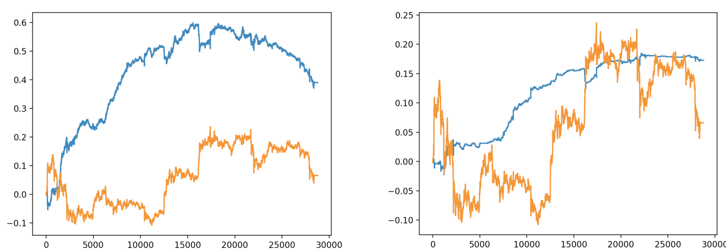


Figure 16: 0.5 Left, 0.6 Right

6.3.5 Theoretical - 2 bips loss/trade

If we could theoretically achieve a loss of 2 bips per trade performance skyrockets. Please keep in mind these are theoretical Sharpes.

Returns & Sharpe with a fee of 0.2¢ per trade on every $10 invested:

	0.5 Thresh	**0.6 Thresh**
Returns:	89.24%	24.54%
Sharpe:	4.12	11.31

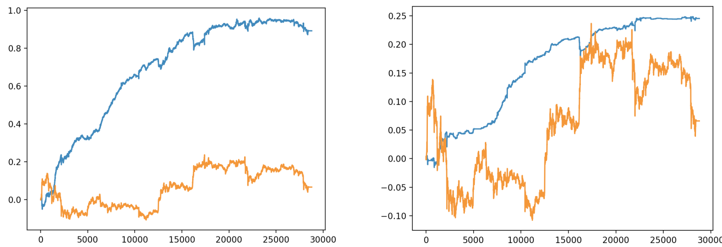


Figure 17: 0.5 Left, 0.6 Right

6.3.6 Theoretical Performance Limit - 0 bips loss/trade

If we could theoretically achieve a loss of 0 bips per trade we can fully separate the algorithm's prediction abilities from its market entry effectiveness. These are the theoretical limits of the Sharpe in a perfect investment state.

Returns & Sharpe with a fee of 0.0¢ per trade on every $10 invested:

	0.5 Thresh	**0.6 Thresh**
Returns:	139.48%	31.80%
Sharpe:	6.44	14.64

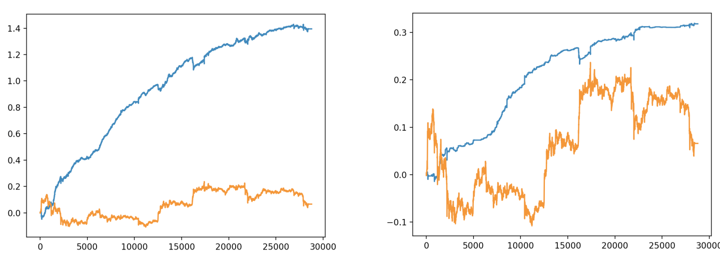


Figure 18: 0.5 Left, 0.6 Right

6.4 Conclusions

The insights gleaned from an analysis of this model & its performance vs. the market holds many conclusions. I discuss three of them here.

First, by intelligently investing on only a subset of potential trades we were able to more than double the Sharpe. This choice demonstrates the effectiveness of the AVA system at reducing volatility.

Second, the performance limit on this strategy is very high. When just analyzing the prediction technology there is lots of potential.

Third, even if you can predict the future that doesn't mean you can profit off those predictions. Section 6.3 illustrates the massive effect even slight changes in fees make to a quant strategy which trades frequently. Notice a one bip fee drop from 5 to 4 doubles the returns on the 0.5 strategy.

7 Trading

7.1 Implementation Approaches

The output of AVA so far is an insightful prediction between (0,1). Granted we know how to predict market quite accurately on a subset of these predictions, however, they still remain predictions. Binary buy & sell at a certain threshold is not necessarily the best way to convert these predictions into alpha. The natural next question is "how do you most effectively trade on these predictions?" In this section I overview the live trading algorithm and offer a few trading strategies I especially like.

7.2 Implementation Approaches

There are theoretically an infinite number of trading strategies which can be superimposed on top of AVA's prediction set. I outline three here.

7.2.1 Binary Buy Sell

Binary Buy Sell is perhaps the simplest of all prediction based trading strategies. This strategy is the one used in section 6.3. If the prediction value is above or below a certain threshold, invest or short accordingly. If not, hold in a stable currency. The returns from this strategy are solid. It can only go up from here.

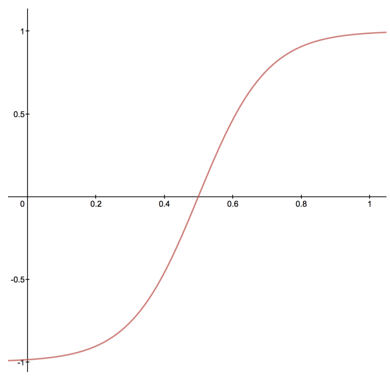
$$f(x) = \begin{cases} -100\% & x < 0.4 \\ 0 & 0.4 \leq x \leq 0.6 \\ 100\% & 0.6 < x \end{cases}$$

7.2.2 Dynamic Asset Scaling

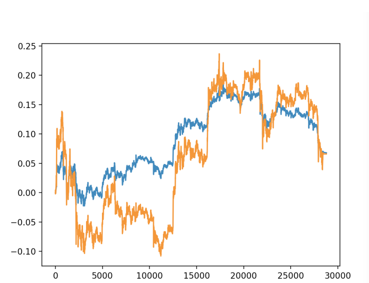
Dynamic Asset Scaling is a slightly more sophisticated version of Binary Buy Sell. Instead of buying or selling as a discrete function of price it scales the total assets held in crypto to factor in the unusual distribution of predictions.

$$f(x) = \frac{2 \cdot e^{5(x-0.5)}}{e^{5(x-0.5)} + e^{-5(x-0.5)}} - 1$$

This function graphed appears as follows, where the x axis is the prediction output, and the y axis the percent of the portfolio held in crypto (negative is short):

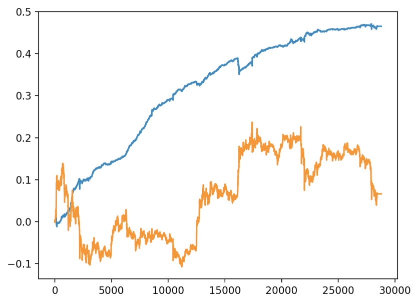


With shorting the strategy performs relatively well, as a near intermediary between the 0.5 and 0.6 Binary Buy Sell strategy. The Sharpe's are 7.23 with no fees and 3.92 with 4 bips/trade fees. The first picture is the strategy without the ability the short, which yields a Sharpe of 0.605.



8 Applications

Today AVA is designed to actively trade on crypto markets. However, this application of the technology is singular, and not holistic. Recall that AVA is simply a robust set of prediction technologies. There are many ways predicting the future can be valuable.



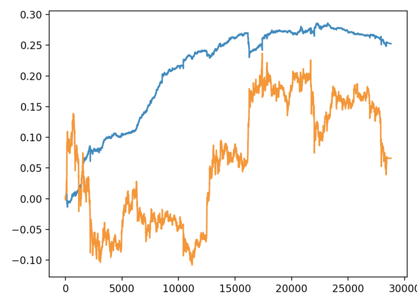


Figure 19: Left: No fees, Right: 4 bips/trade fee

I see AVA assisting other quant shops as an input into other trading strategies.

AVA can serve as the "belly" of a reinforced learning algorithm designed for pure alpha.

A fundmental trader can use AVA's predictions to decide when the best time to open a position.

AVA can be used as a dashboard for traders to passively monitor where markets are headed. I see a wall of TV screens showing where the market is probabilistically heading in n minutes into the future.

I can see a reality where better models are subbed in and AVA becomes an even stronger prediction mechanism.

Someone who has a great market entry algorithm can utilize AVA to make money directly off their technology.

AVA based crypto assets could be bundled into the world's first truly efficient crypto ETF, solely consisting of high grade coins.

This list is a brainstorm of what I have assembled so far. I'm excited to see where AVA goes next.

References

[1] William L. Hamilton, Kevin Clark, Jure Leskovec, Dan Jurafsky. *Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora.* Con-

ference on Empirical Methods in Natural Language Processing. 2016. https://arxiv.org/pdf/1606.02820.pdf