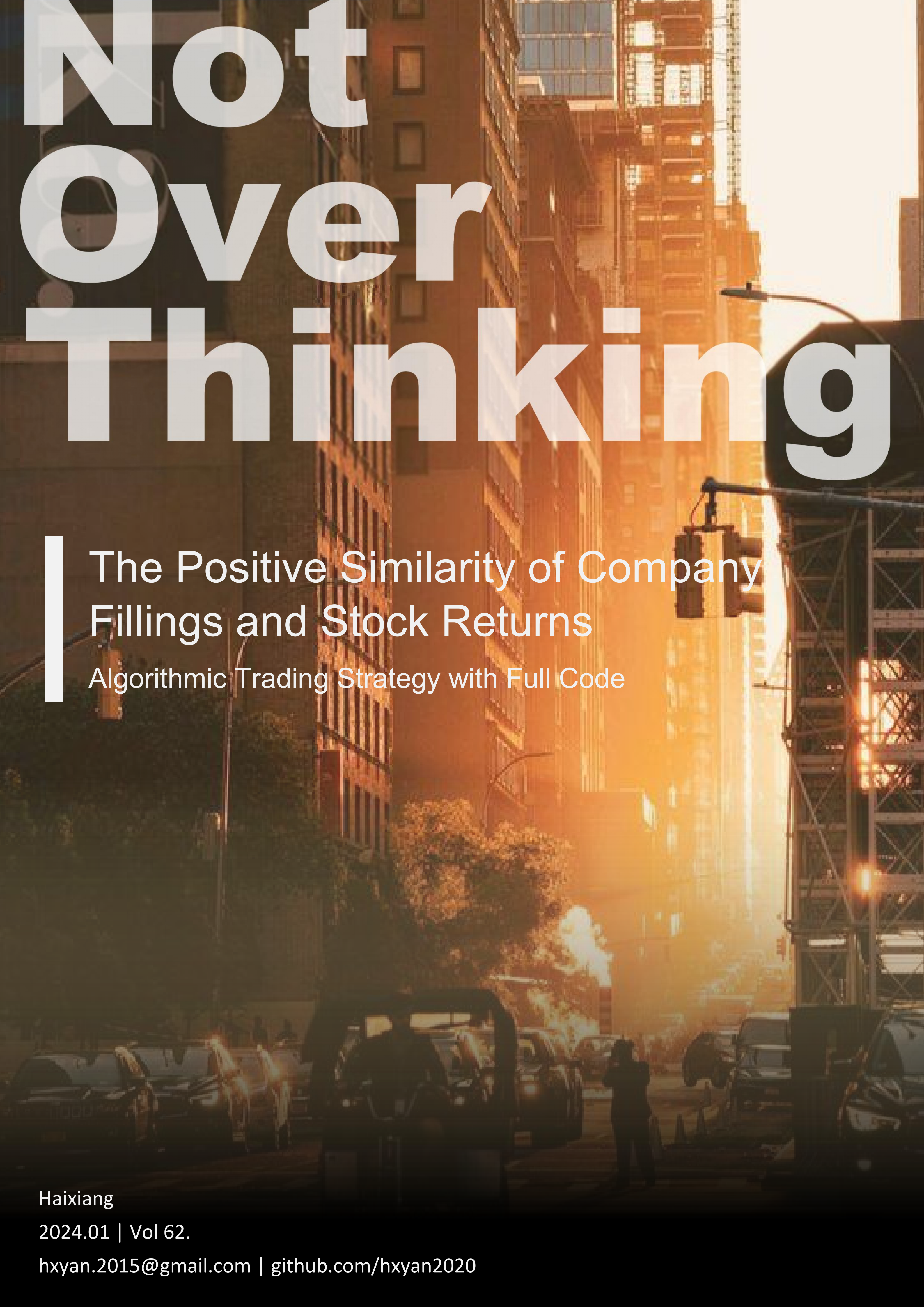


Not Over Thinking

The Positive Similarity of Company Fillings and Stock Returns

Algorithmic Trading Strategy with Full Code

Haixiang

2024.01 | Vol 62.

hxyan.2015@gmail.com | github.com/hxyan2020

STRATEGY & ECONOMIC RATIONALE

The investment universe consists of stocks with large market cap covered by the Brain Company, for which stock prices were available to download from Yahoo Finance and had full history during the sample period. Firstly, only the similarity of the positive language is considered. The positive similarity score is calculated as the cosine similarity and is provided by the Brain Company. Each month, stocks are ranked based on the positive similarity language score of their most recent company filing and sorted into deciles. Long the bottom decile and short the top decile. The strategy is equally-weighted and rebalanced monthly.

BUY	SELL
Long the bottom decile	short the top decile

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Equity
FINANCIAL INSTRUMENTS	Stocks
REGION	United States
PERIOD OF REBALANCING	Monthly
NO. OF TRADED INSTRUMENTS	637
WEIGHTING	Equal weightin
LOOKBACK PERIODS	N/A
LONG/SHORT	Long & Short

ALGORITHM

```
from AlgorithmImports import *# endregion
class ThePositiveSimilarityOfCompanyFilingsAndStockReturns(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2009, 1, 1) # first metric data come in 2009
        self.SetCash(100000)

        self.leverage:int = 5
        self.quantile:int = 10

        self.metric_symbols:dict[Symbol, Symbol] = {}
        self.positive_similarities:dict[Symbol, float] = {}

        self.market:Symbol = self.AddEquity('SPY', Resolution.Daily).Symbol

        self.coarse_count:int = 1000
        self.pick_largest:bool = True

        self.selection_flag:bool = False
        self.UniverseSettings.Resolution = Resolution.Daily
        self.AddUniverse(self.CoarseSelectionFunction, self.FineSelectionFunction)
        self.Schedule.On(self.DateRules.MonthStart(self.market), self.TimeRules.BeforeMarketClose(self.market, 0), self.Selection)

    def OnSecuritiesChanged(self, changes):
        for security in changes.AddedSecurities:
            security.SetFeeModel(CustomFeeModel())
            security.SetLeverage(self.leverage)

    def CoarseSelectionFunction(self, coarse):
        if not self.selection_flag:
            return Universe.Unchanged

        if self.coarse_count <= 1000 and not self.pick_largest:
            selected:list = sorted([x for x in coarse if x.HasFundamentalData and x.Market == 'usa'],
                    key=lambda x: x.DollarVolume, reverse=True)[:self.coarse_count]
        else:
            selected:list = [x for x in coarse if x.HasFundamentalData and x.Market == 'usa']

        selected_symbols:list[Symbol] = []

        for stock in selected:
            symbol:Symbol = stock.Symbol

            if symbol not in self.metric_symbols:
                metric_symbol:Symbol = self.AddData(BrainCompanyFilingLanguageMetrics10K, symbol).Symbol
                self.metric_symbols[symbol] = metric_symbol

            selected_symbols.append(symbol)

        return selected_symbols

    def FineSelectionFunction(self, fine):
        if self.coarse_count <= 1000:
            return list(map(lambda stock: stock.Symbol, fine))

        fine:list = list(filter(lambda stock: stock.MarketCap != 0, fine))

        if len(fine) > self.coarse_count or self.pick_largest:
            sorted_by_cap:list = sorted(fine, key=lambda stock: stock.MarketCap)
            fine = sorted_by_cap[-self.coarse_count:]

        return list(map(lambda stock: stock.Symbol, fine))

    def OnData(self, data):
        if self.selection_flag:
            self.selection_flag = False

            filtered_positive_similarity:dict[Symbol, float] = { symbol: pos_sim for symbol, pos_sim in self.positive_similarities.items() \
                if symbol in data and data[symbol] }

            self.positive_similarities.clear()

            if len(filtered_positive_similarity) < self.quantile:
                self.Liquidate()
            else:
                quantile:int = int(len(filtered_positive_similarity) / self.quantile)
                sorted_by_pos_sim:list[Symbol] = [x[0] for x in sorted(filtered_positive_similarity.items(), key=lambda item: item[1])]

                long_leg:list[Symbol] = sorted_by_pos_sim[:quantile]
                short_leg:list[Symbol] = sorted_by_pos_sim[-quantile:]

                invested:list[Symbol] = [x.Key for x in self.Portfolio if x.Value.Invested]
                for symbol in invested:
                    if symbol not in long_leg + short_leg:
                        self.Liquidate(symbol)

                for symbol in long_leg:
                    self.SetHoldings(symbol, 1 / quantile)

                for symbol in short_leg:
                    self.SetHoldings(symbol, -1 / quantile)

        for stock_symbol, metric_symbol in self.metric_symbols.items():
            if metric_symbol in data and data[metric_symbol]:
                positive_similarity:float = data[metric_symbol].ReportSentiment.Similarity.Positive

                if positive_similarity:
                    self.positive_similarities[stock_symbol] = positive_similarity

    def Selection(self):
        self.selection_flag = True
# Custom fee modelclass CustomFeeModel(FeeModel):
    def GetOrderFee(self, parameters):
        fee = parameters.Security.Price * parameters.Order.AbsoluteQuantity * 0.00005
        return OrderFee(CashAmount(fee, "USD"))
```

BACKTESTING PERFORMANCE

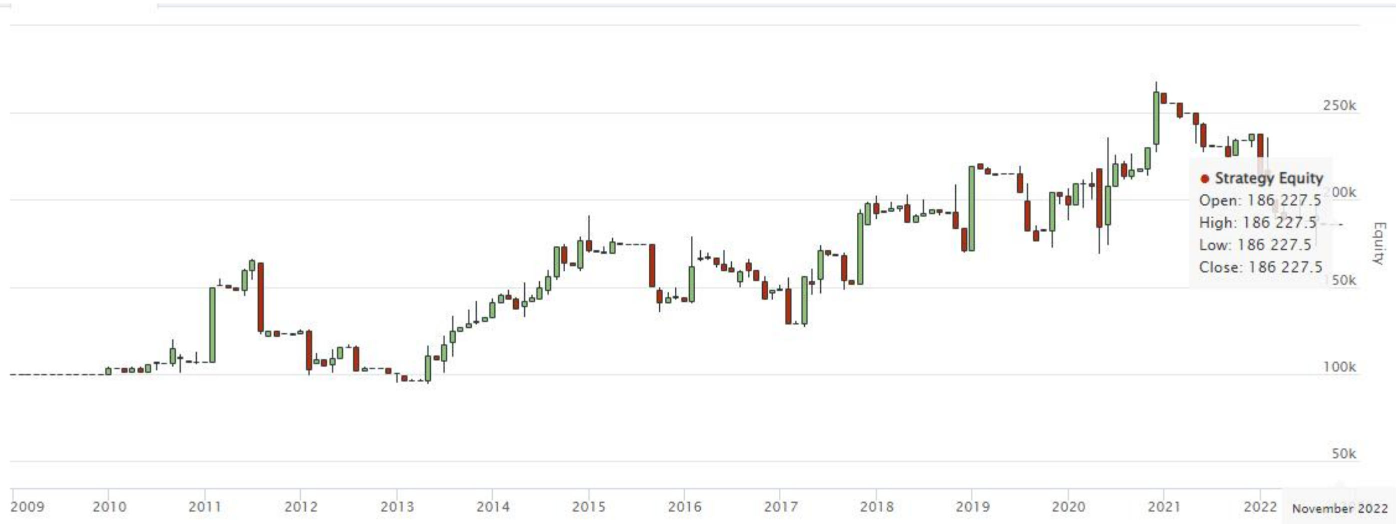
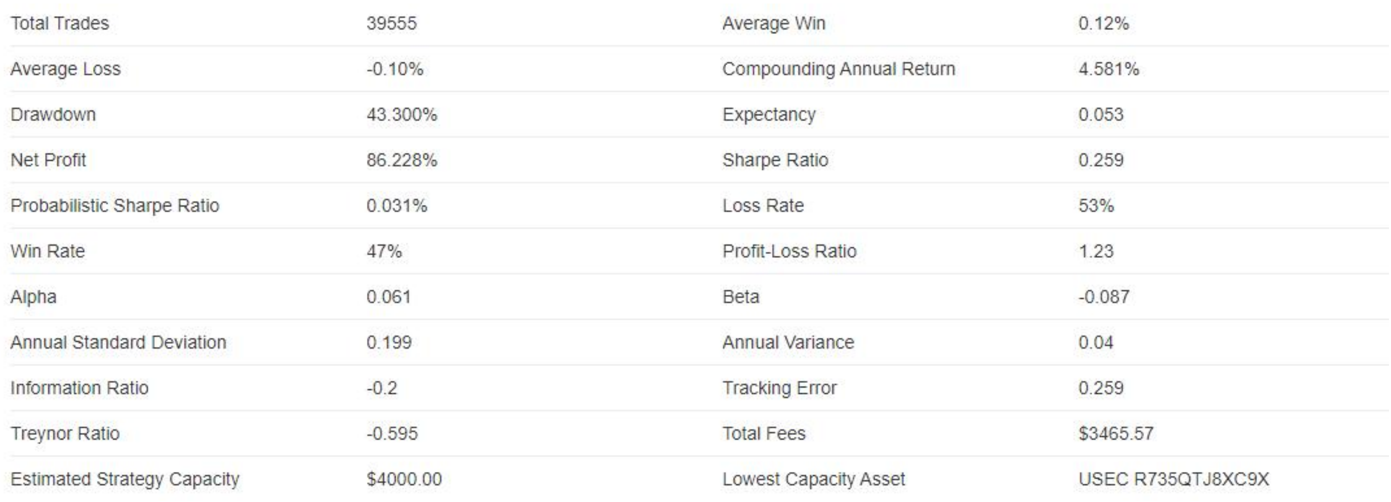


Fig 1. Overall Performance



Total Trades	39555	Average Win	0.12%
Average Loss	-0.10%	Compounding Annual Return	4.581%
Drawdown	43.300%	Expectancy	0.053
Net Profit	86.228%	Sharpe Ratio	0.259
Probabilistic Sharpe Ratio	0.031%	Loss Rate	53%
Win Rate	47%	Profit-Loss Ratio	1.23
Alpha	0.061	Beta	-0.087
Annual Standard Deviation	0.199	Annual Variance	0.04
Information Ratio	-0.2	Tracking Error	0.259
Treynor Ratio	-0.595	Total Fees	$3465.57
Estimated Strategy Capacity	$4000.00	Lowest Capacity Asset	USEC R735QTJ8XC9X

Fig 2. Performance Metrics