Data Science Institute
COLUMBIA UNIVERSITY

## Fall 2020 Capstone Project

## Progress Report 2

# Measuring startup strategy and its evolution

**November 24, 2020**


### Members

Derek Chen - cc4506
Wangzhi Li - wl2737
Bernardo Lopez Vicencio - bl2786
Yinhe Lu - yl4372
Alberto Munguia Cisneros - am5334

### Mentor

Prof. Jorge Guzman
Columbia Business School

# Table of Contents

# I. Introduction

This second progress report of our project is devoted to tackling some relevant tasks that will allow us to achieve the goal of our project - developing a new analysis of the strategy of firms using text-based machine learning.

On the web-scraping line of work, we implemented an algorithm to collect on a large scale and in a reasonable amount of time all the first level data in the web-pages from the entire universe of startups and public companies in 20 years. Furthermore, we implemented another cleaning and processing filter to assure the quality and consistency of our data before the modeling phase. Regarding the NLP embedding models phase, we implemented Bidirectional Encoder Representations from Transformers model (BERT) and compared its performance in terms of our strategy score with the techniques NLP embeddings techniques that we have already implemented TF-IDF and Word2Vec. In the last section of this progress report, we presented the results of the regression that we implemented to measure the relation between the strategy score and the performance of our startup sample.

# II. Multithreading for Web Scraping and Data Processing

## A. Web scraping

Due to the large amount of startup web data that needed to scrape to perform our prediction analysis, we devoted a significant amount of time and resources to the data collection process. The main hurdle that we encountered was to improve our initial version of our web-scraping algorithm to make it more efficient in time and memory use.

Our initial version web-scraping algorithm allowed us to effectively web scrape the first level information of the historical website for public and startups from Wayback Machine internet archive. The average time to web scrape the first level[1] data for a company was on average 46 seconds, considering an EC2 Machine on AWS with 12GB of RAM. Although this performance appears adequate, it was insufficient considering the universe of companies that we need to web scrape in a 20-year window of time; the estimated time to web scrape the entire universe of companies considering our initial resource was 218.6 days of continuous processing.

*Table 1 Time estimation for web scraping*

| Type | # companies | Estimated time processing(hrs) | Estimated time processing (days) |
|---|---|---|---|
| Startups | 13,677 | 174.8 | 7.3 |
| Public Companies | 396,896 | 5,071.4 | 211.3 |
| **Total** | **410,573** | **5,246** | **218.6** |

[1] First Level data corresponds to all the information contained in the links of the homepage website.

To overcome this potential bottleneck that could jeopardize the success of our project, we implemented a dual solution. First, we increase our hardware resources. We decided that an easy and cheap option to achieve this purpose was to upgrade our subscription to Google Colab, which allows us the concurrent use of 4 virtual machines with extended RAM, 25GB, and to store our results directly in a Google drive. Second, we implemented the python library 'concurrent.futures', which is a multithreading technique that allowed us to update our scraping code and achieve a performance 20 times faster than the initial version.

In the concurrent.futures library, the python engine allows concurrent execution of I/O tasks through multithreading. This functionality enabled us to leverage a ThreadPoolExecutor to perform scraping through multiple threads. This I/O functionality releases the Global Interpreter Lock, which means our scraping process is executed concurrently across different threads. The asynchronous processing constituted a clear advantage to our initial process that relied on a sequential process to perform the web scraping.

*Table 2 Performance improvement comparison*

| Size sample | Estimated time process(hrs) | | Performance (times) |
|---|---|---|---|
| | Initial | Multithreading | |
| 200 | 2.6 | 0.6 | 4.6 |
| 1,500 | 19.2 | 0.9 | 21.7 |

The upgrade in hardware resources and the multithreading version of our web scraping code allowed us to finish the data collection of the entire universe of companies in two weeks.

## B. Data processing

The output produced in the web scraping process is a collection of text files, one for each company. However, the next steps of our process require that data should be as consistent and clean as possible and structured in a data frame. To achieve these purposes, we implemented the following steps.

- **Data structure**: Opening and reading all output files sequentially requires a considerable amount of time. So, to improve the efficiency of our processes, we decided to concentrate all text in one single CSV file for each year. To speed up this process, we used a multithreading process from the python library concurrent.futures.

- **Language Consistency**: We found out that some websites had text in different languages. This inconsistency could generate an error in the NLP embeddings algorithms. So, to get a homogeneous sample, we implemented a filter to detect the language and filter those texts that were not in English. To carry on this task, we used the Compact Language Detector v3 from Google which is a pre-trained neural network model for language identification.

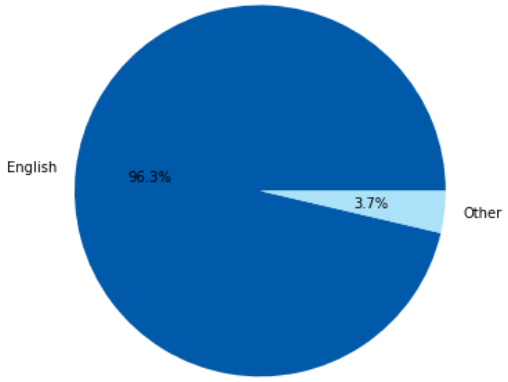



*Fig1: Percentage of websites in English*

- **Outliers**: We found out that for some companies the web scraping process produces text with more than 4 million words. Analyzing these extremely large texts proved to be inefficient because just loading the CSV file with these websites caused our Google Colab instances to run out of RAM. To avoid this problem, we decided to drop all companies whose website produces a text with more than 1 million words. By dropping these companies, we dropped less than 0.5% of the universe of companies

  The following Table 3 is a summary of the number of websites web scraped per year.

*Table 3 Counts of web scraped companies*

| Year | Total # of companies | # of startups | # of public companies | # of websites in english | # of websites in other languages |
|---|---|---|---|---|---|
| **2000** | 157 | 49 | 108 | 146 | 11 |
| **2001** | 3975 | 51 | 3924 | 3755 | 220 |
| **2002** | 1722 | 44 | 1678 | 1642 | 80 |
| **2003** | 4656 | 104 | 4552 | 4404 | 252 |
| **2004** | 4655 | 142 | 4513 | 4457 | 198 |
| **2005** | 3680 | 147 | 3533 | 3562 | 118 |
| **2006** | 5357 | 181 | 5176 | 5202 | 155 |
| **2007** | 5656 | 221 | 5435 | 5475 | 181 |
| **2008** | 5237 | 211 | 5026 | 5068 | 169 |

| | | | | | |
|---|---|---|---|---|---|
| **2009** | 5390 | 294 | 5096 | 5215 | 175 |
| **2010** | 2149 | 430 | 1719 | 2053 | 96 |
| **2011** | 6304 | 539 | 5765 | 6137 | 167 |
| **2012** | 6744 | 768 | 5976 | 6524 | 220 |
| **2013** | 7331 | 894 | 6437 | 7106 | 225 |
| **2014** | 7623 | 1014 | 6609 | 7334 | 289 |
| **2015** | 7815 | 1079 | 6736 | 7532 | 283 |
| **2016** | 6803 | 885 | 5918 | 6549 | 254 |
| **2017** | 7442 | 842 | 6600 | 7163 | 279 |
| **2018** | 5361 | 507 | 4854 | 5160 | 201 |
| **2019** | 5676 | 231 | 5445 | 5457 | 219 |

# III. Model Comparison TF-IDF vs Word2Vec vs BERT

This section is devoted to the description of our third NLP embedding technique and the comparison of the versions of the strategy score that we obtained from the NLP techniques that we implemented TF-IDF, Word2Vec and BERT.

## A. BERT implementation

One of the pending lines of research regarding the NLP embedding techniques was the implementation of an alternative algorithm to our initial choices, TF-IDF and Word2Vec. For this purpose, we implemented BERT or Bidirectional Encoder Representations from Transformers which is a Transformer-based machine learning technique for natural language processing (NLP) pre-training and developed by Google. BERT is designed to pre-train deep bidirectional representations from an unlabeled text by jointly conditioning on both left and right context in all layers. BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text.

## B. Strategy Score comparison

Given the constraints of time and resources available, we headed to compare strategy scores[2] issued form our three embedding techniques in 2011. Also, our analysis on the data from 2012 produced a very similar result as the one from 2011 and therefore we only introduce our findings in 2011 here.

[2] Strategy Score is the average the distance among the closest statements for each startup. Where cosine similarity is the metric for measuring the text similarity.

We are trying to compare the differences in distribution of the strategy scores from these three models. We used the three different approaches to calculate the similarity between every two companies. Based on that, we can get the strategy scores furthermore.

First, we calculate the mean value and the variance for the strategy scores. For mean values, if-idf has the highest mean values(0.833)versus word2vec has the lowest mean values(0.045). For variance, TF-IDF has a relatively higher variance than the others, and word2vec and Bert have a relatively smaller variance.

*Table 4 Mean and Variance for all strategy scores in 2011*

| | *TF-IDF* | *word2vec* | *bert* |
|---|---|---|---|
| *Mean* | *0.833* | *0.045* | *0.305* |
| *Variance* | *0.013* | *0.001* | *0.003* |

Second, we compared the frequency of the strategy scores for all these startups. As we can see in these two graphs, TF-IDF tends to give a higher score whereas word2vec tends to give a low score. The distribution of TF-IDF is more spread-out than the others. But the question we still need to ask ourselves is "does the frequency distribution mean these three models give totally different results?" We need to delve into the results further so that we can verify them.

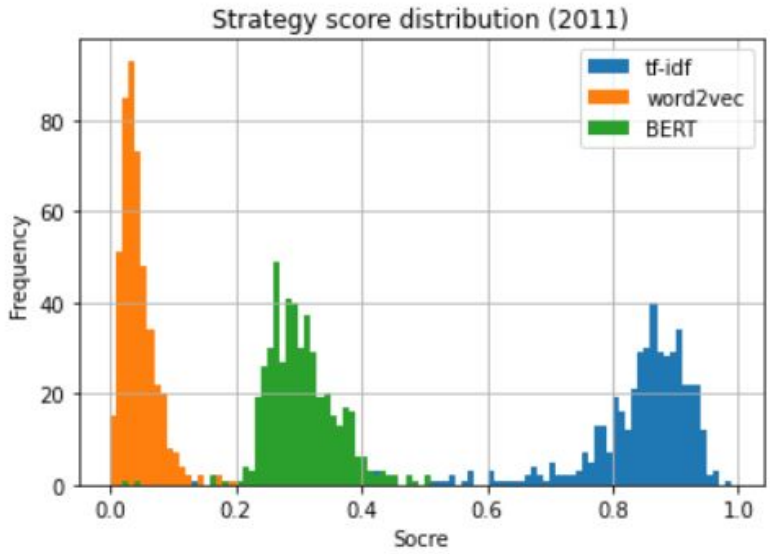



*Figure 2: Distributions for Strategy Score for three models in 2011*

Third, in order to check if the three models have distinction, we checked correlations for the results for these three models. If the strategy scores are correlated to each other for all these three models, that means we actually have pretty similar evaluation of strategy scores regardless of what method we used.

So we plotted the scatter plots and calculated the Pearson correlations. From Figure 3, despite the different distributions, the scatter plots show that they are actually linearly correlated. Moreover, we calculated the Pearson correlation coefficient and the p values and presented in Table 5, the strategy scores are significantly correlated to each other, which means from this perspective, we have similar results.

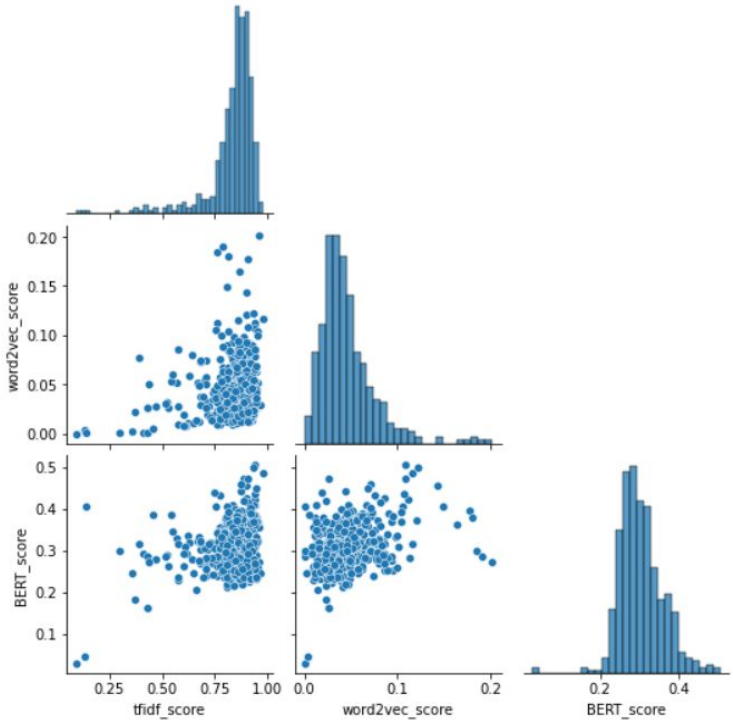



*Figure 3: Correlations for Strategy Score for three models in 2011*

*Table 5: Pearson Correlations for three models in 2011*

| | *TF-IDF* | *word2vec* | *bert* |
|---|---|---|---|
| *TF-IDF* | - | *0.22**** | *0.26**** |
| *word2vec* | *0.22**** | - | *0.36**** |
| *bert* | *0.26**** | *0.36**** | - |

# IV. Statistical Analysis - Strategy vs Performance

## A. Defining performance

To measure the success of a company, we used two different sources of information:

1. All deals database: This database contained information about all investments received by a company. We had the type of investment (seed funding, round number, etc.), the amount invested and the date.

2. Preqin exits database: Contains detailed information about each exit of a company. The exit types are IPO, Merger, Private Placement, Recapitalisation, Restructuring, Sale to GP, Sale to Management, and Trade Sale. However, for this analysis, we only used IPO and Trade Sale because these last two better capture the return on investment for the founders of a startup.

Using these two sources of information, we built the following performance variables for each startup:

- Deals count: Total number of deals (investments) received by a startup so far.
- Deal size: Sum of the value of all deals (investments) received by a startup so far. This quantity is in millions.
- Seed funding: Million of dollars invested in seed funding.
- First rounds funding: Million of dollars invested in the seed funding and in the first four rounds (i.e. Series A, Series B, Series C, and Series D). As suggested by Figure 4, each round of funding has about 12 months of interval, during which strategies of startups might change. Therefore, we only select seed and the first four rounds of funding.
- Has Seed: Binary variable that indicates if a startup has seed funding.
- Has IPO: Binary variable that indicates if a startup has IPO.
- IPO value: Value of the company (in millions of dollars) when it IPO
- Has trade sale: Binary variable that indicates if a startup has an exit of type “Trade Sale”. These are startups that have been sold.
- Trade Sale value: Value of the company (in millions of dollars) when it was sold.

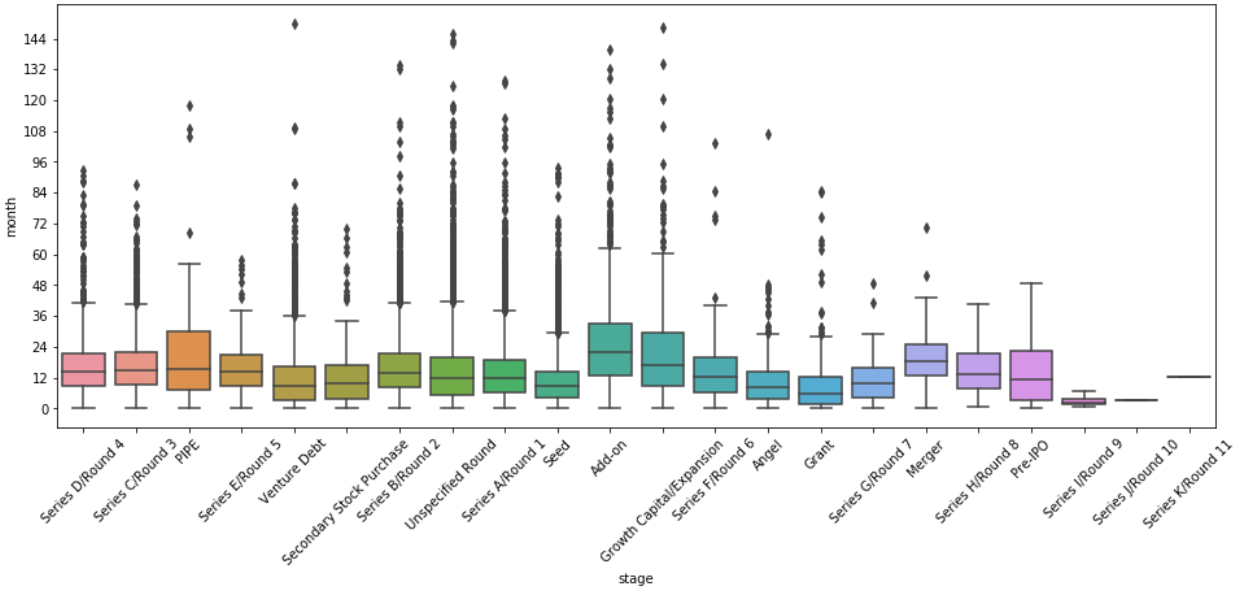



*Figure 4: Box plot of time intervals between each round of funding and its previous one (month)*

## B. Linear Regression Analysis

We now proceed to estimate the startup founding strategy and its relationship to startup performance. As mentioned in the previous section, we applied three kinds of text similarity analysis techniques to calculate the strategy score. To evaluate the effectiveness of proposed strategy scores,

linear regression models are used to investigate the relationship between strategy scores and dependent variables that could describe the performance of startups.

It is important to note that startups' strategies are possible to evolve through the time to adapt to new situations. In this sense, although it is reasonable to assume that their founding strategy might help them to get future success, the correlation between strategy scores we calculated and dependent variables such as 'has IPO' would be smaller than variables such as 'seed funding'. Therefore, here we only select two representative cases, "seed funding" and "has IPO", to introduce our work. Further information can be found in the Appendix.

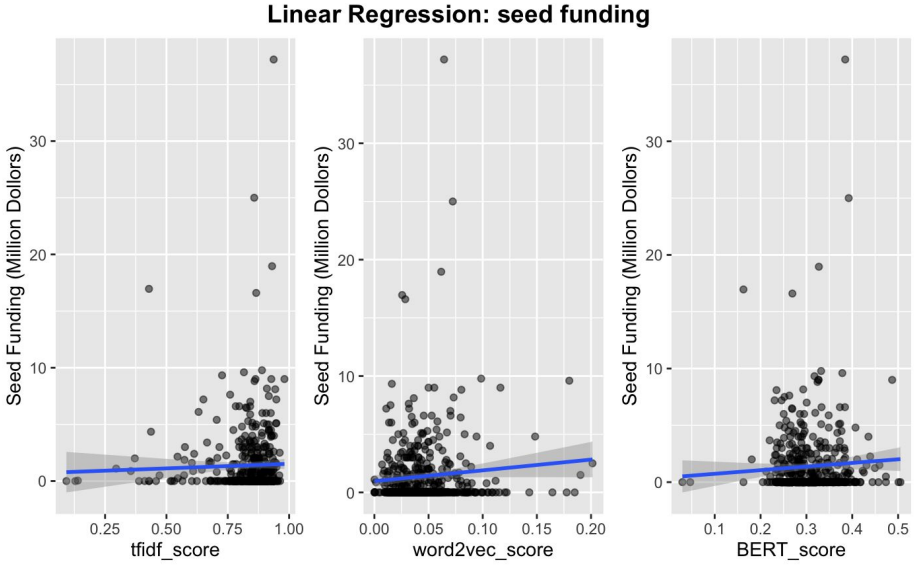



*Figure 7: Linear regression models of founding strategy scores and seed funding*

*Table 8: Founding strategy scores and seed funding*

**Linear Regression: seed funding**

| | *Dependent variable:* | | |
|---|---|---|---|
| | seed_founding | | |
| | (1) | (2) | (3) |
| tfidf_score | 0.813 | | |
| | (1.213) | | |
| word2vec_score | | 9.255* | |
| | | (4.907) | |
| BERT_score | | | 3.190 |
| | | | (2.560) |
| Constant | 0.710 | 0.969*** | 0.415 |
| | (1.021) | (0.264) | (0.793) |
| Observations | 470 | 470 | 470 |
| $R^2$ | 0.001 | 0.008 | 0.003 |
| Adjusted $R^2$ | -0.001 | 0.005 | 0.001 |
| Residual Std. Error (df = 468) | 3.095 | 3.084 | 3.091 |
| F Statistic (df = 1; 468) | 0.449 | 3.558* | 1.553 |
| *Note:* | *p<0.1;* ***p<0.05;*** p<0.01 | | |

From both Figure 7 and Table 8, we can find positive relationships between seed funding value and our proposed strategy scores, which indicates that with a higher strategy score, a startup is more

likely to get more seed funding. However, only the wor2vec-based strategy score shows a statistically significant positive slope and all three linear regression models get a almost zero R2 value. This could be partially explained by influence of other confounding factors, such as geographic location and industry.

For startups' performance in a long period, our strategy score based on Word2Vec shows a positive correlation with the binary dependent variable, has IPO, which indicates that with a higher Word2Vec-based strategy score, the start up is more likely to have IPO. Nonetheless, such a conclusion is not demonstrated by strategy scores based on TF-IDF or BERT. Note that the reason why we use “has IPO” rather than “IPO value” is because the highly skewed IPO value might add unnecessary noise to our model and going IPO itself can effectively show the success of a startup as well.

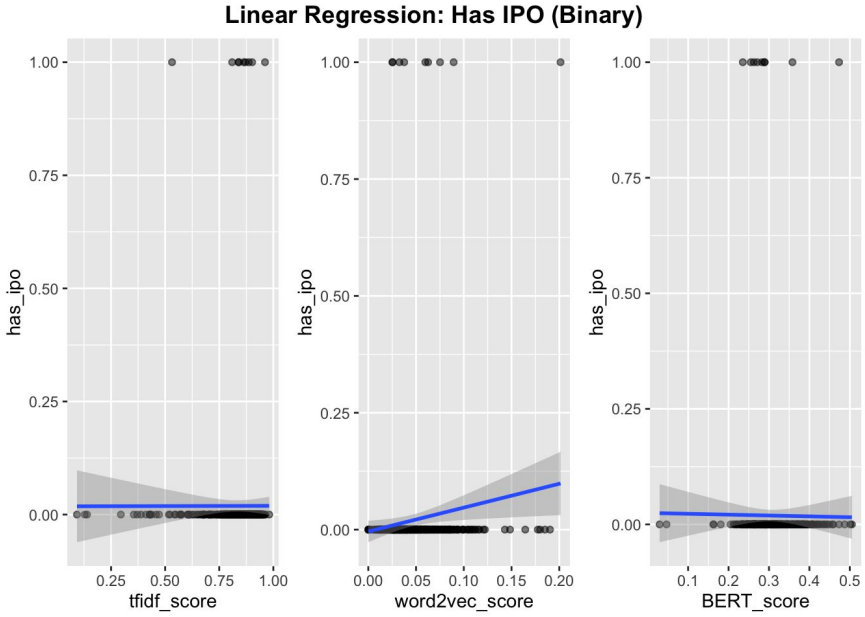



*Figure 8: Linear regression models of founding strategy scores and has IPO*

*Table 9: Founding strategy scores and has IPO*

**Linear Regression: Has IPO (binary)**

| | *Dependent variable:* | | |
|---|---|---|---|
| | has_ipo | | |
| | (1) | (2) | (3) |
| tfidf_score | 0.001 | | |
| | (0.054) | | |
| word2vec_score | | 0.510** | |
| | | (0.217) | |
| BERT_score | | | -0.019 |
| | | | (0.114) |
| Constant | 0.018 | -0.004 | 0.025 |
| | (0.045) | (0.012) | (0.035) |
| Observations | 470 | 470 | 470 |
| $R^2$ | 0.00000 | 0.012 | 0.0001 |
| Adjusted $R^2$ | -0.002 | 0.010 | -0.002 |
| Residual Std. Error (df = 468) | 0.137 | 0.137 | 0.137 |
| F Statistic (df = 1; 468) | 0.0004 | 5.520** | 0.027 |
| *Note:* | *p<0.1;* ***p<0.05;*** p<0.01 | | |

## V. Next Steps

Our main task in the final stage is to improve the design of strategy score as well as to make comprehensive analyses on our proposed strategy scores through linear regression modelling. Specifically, our plan is as follows.

1. Build up fixed-effect models to investigate the influence of confounding variables. Fixed-effect models will address the influence of other confounding variables, for example industry of startup. These variables will partially explain the dependent variable, which is arguably one of the reasons for the low R2 value in our current linear regression model. We would like to take into consideration geographical location, industry, and founding year.
2. Calculate strategy score based on similarity between both public companies and other startups. The design of strategy score is based on the assumption that less similarity in positioning would indicate more market space. However, competitors of startups are not only public companies but also other startups. To take their positioning into consideration would make our strategy score more reasonable. Furthermore, we also need to consider the weight of positioning of startups and public companies in the design of our strategy score.
3. Combine data from multiple years to calculate strategy scores. Currently, we only made an end-to-end analysis for one year, 2011, which might lead us to biased conclusions. To mitigate this effect, we need to upscale our experiments and combine all of our data over a period of two decades to build linear regression models.

## Github

The code for all the sections of the project is available in a Github repository https://github.com/derekcoding1/StartupStrategy

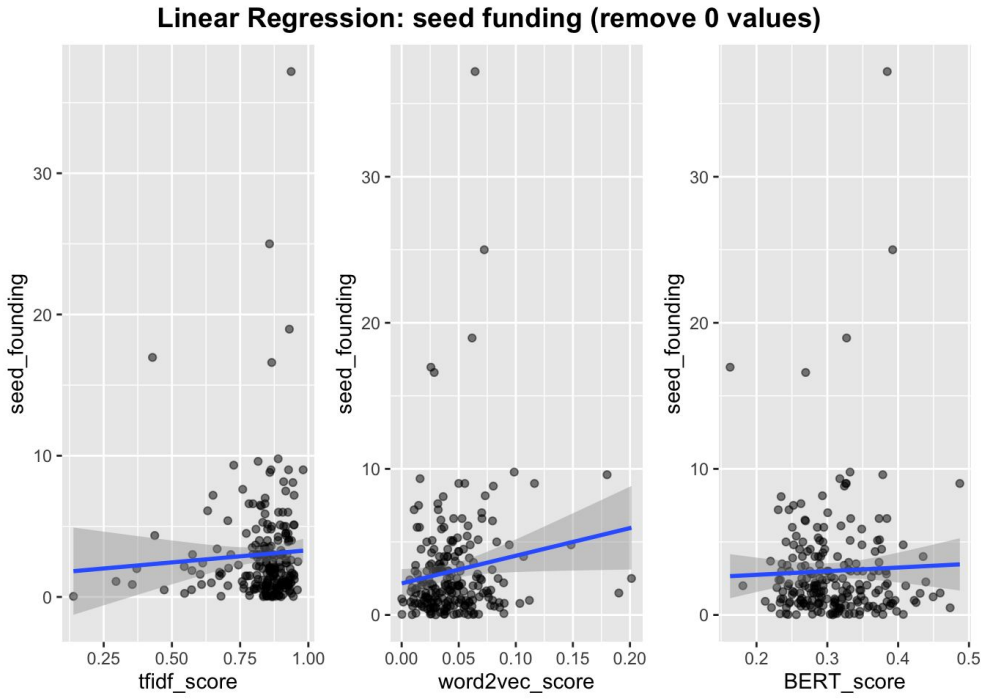
# Appendix

*Appendix Table 1: Founding strategy scores and seed funding(remove 0 values)*

**Linear Regression: seed funding (remove 0 values)**

| | *Dependent variable:* | | |
|---|---|---|---|
| | seed_founding | | |
| | (1) | (2) | (3) |
| tfidf_score | 1.723 | | |
| | (2.225) | | |
| word2vec_score | | 18.924** | |
| | | (9.150) | |
| BERT_score | | | 2.492 |
| | | | (4.873) |
| Constant | 1.585 | 2.160*** | 2.249 |
| | (1.872) | (0.495) | (1.532) |
| Observations | 216 | 216 | 216 |
| $R^2$ | 0.003 | 0.020 | 0.001 |
| Adjusted $R^2$ | -0.002 | 0.015 | -0.003 |
| Residual Std. Error (df = 214) | 3.993 | 3.959 | 3.996 |
| F Statistic (df = 1; 214) | 0.600 | 4.278** | 0.262 |
| *Note:* | *p<0.1;* ***p<0.05;*** p<0.01 | | |

*Appendix Figure 1: Founding strategy scores and seed funding(remove 0 values)*

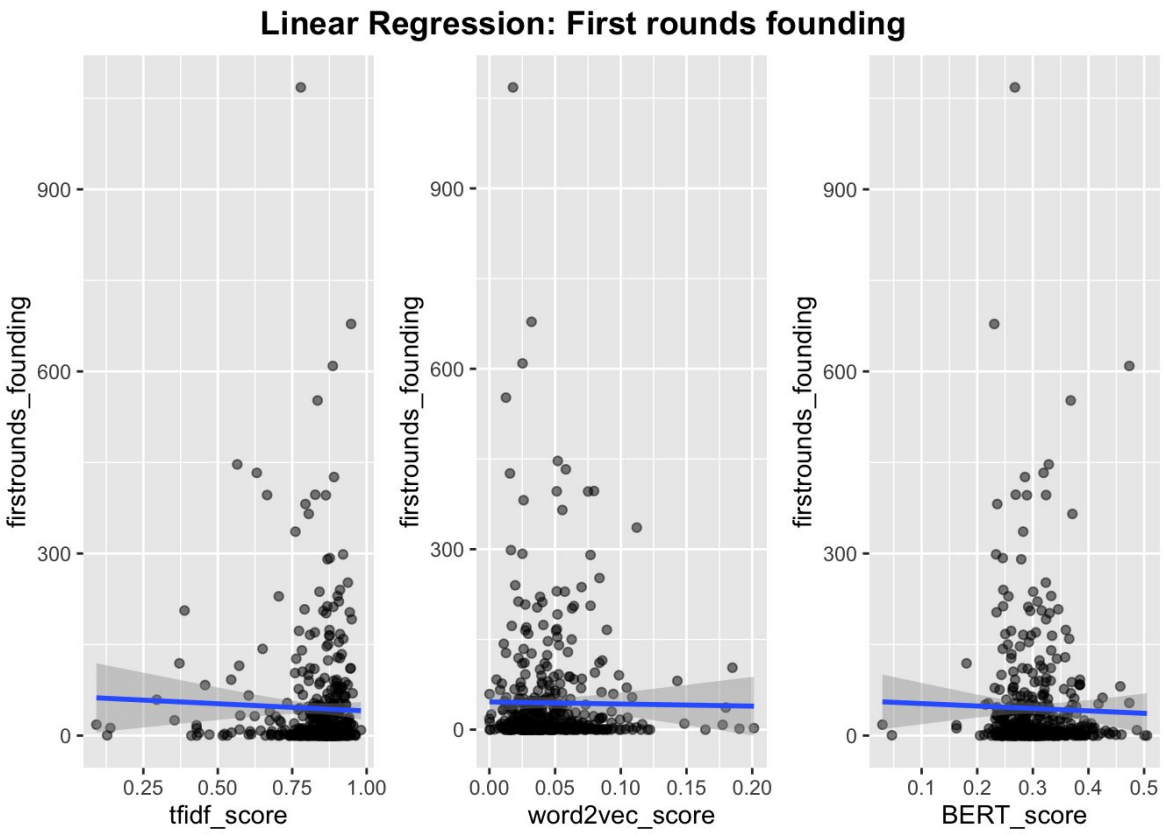


*Appendix Table 2: Founding strategy scores and first rounds funding*

**Linear Regression: First rounds founding**

| | *Dependent variable:* | | |
|---|---|---|---|
| | firstrounds_founding | | |
| | (1) | (2) | (3) |
| tfidf_score | -23.873 | | |
| | (38.661) | | |
| word2vec_score | | -34.947 | |
| | | (156.908) | |
| BERT_score | | | -40.154 |
| | | | (81.668) |
| Constant | 64.400** | 46.093*** | 56.752** |
| | (32.527) | (8.434) | (25.308) |
| Observations | 470 | 470 | 470 |
| $R^2$ | 0.001 | 0.0001 | 0.001 |
| Adjusted $R^2$ | -0.001 | -0.002 | -0.002 |
| Residual Std. Error (df = 468) | 98.596 | 98.631 | 98.610 |
| F Statistic (df = 1; 468) | 0.381 | 0.050 | 0.242 |
| *Note:* | | | *p<0.1; **p<0.05;** p<0.01 |

*Appendix Figure 2: Founding strategy scores and first rounds funding*



*Appendix Table 3: Founding strategy scores and has Trade Sale*

**Linear Regression: Has Trade Sale (binary)**

| | *Dependent variable:* | | |
|---|---|---|---|
| | has_tradesale | | |
| | (1) | (2) | (3) |
| tfidf_score | 0.030 | | |
| | (0.176) | | |
| word2vec_score | | -0.405 | |
| | | (0.715) | |
| BERT_score | | | -0.064 |
| | | | (0.372) |
| Constant | 0.254* | 0.297*** | 0.298** |
| | (0.148) | (0.038) | (0.115) |
| Observations | 470 | 470 | 470 |
| $R^2$ | 0.0001 | 0.001 | 0.0001 |
| Adjusted $R^2$ | -0.002 | -0.001 | -0.002 |
| Residual Std. Error (df = 468) | 0.449 | 0.449 | 0.449 |
| F Statistic (df = 1; 468) | 0.029 | 0.321 | 0.029 |
| *Note:* | *p<0.1;* ***p<0.05;*** p<0.01 | | |

*Appendix Figure 3: Founding strategy scores and has Trade Sale*