B (Online) Appendix: Sample Construction

The amount of data provided by the Spinn3r stream is immense (on average 100GB per day, with some days up to 400GB). Most of it is non-business related content, the typical data available per news post being illustrated in Exhibit 5. To extract business and earnings relevant news from the stream on a daily basis, I proceed in two broad steps. First, I use a dedicated set of keywords to only capture news posts that contain at least three keywords and the name or ticker of a Russel3000 company and save them to a non-relational database. This data constitutes the raw unfiltered data available. The range of selected keywords is broad on purpose to reduce the type II error rate (mistakenly identifying earnings related news as non-earnings news and delete them). Because of this, the type I error rate is quite high (mistakenly identifying non-earnings news as earnings news) after step 1. In a second step, a classification algorithm is trained to identify the non-earnings news and filter them out. After the first step, the raw unfiltered data consists currently 145,234,235 posts from March, 1st 2015 till November 22nd, 2016 for encompassing earnings related news of 2989 firms.

After applying the second step, I end up with the raw dataset used in this study with the characteristics presented in Table 1.

Exhibit 5: A basic example of Spinn3r data in JSON format


```
{
"bucket" : 0,
"resource" : "http://cnn.com/2014/10/15/health/texas-ebola-outbreak",
"date_found" : "2014-06-22T01:08:52Z",
"index_method" : "PERMALINK_TASK",
"html" : "<html><body><p>Full HTML of the content</p></body></html>",
"html_length" : 57,
"source_hashcode" : "COHOcFU4G1sMlRHd9gEvS-n3FFI",
"source_resource" : "http://cnn.com",
"source_link" : "http://cnn.com",
"source_publisher_type" : "MAINSTREAM_NEWS",
"source_date_found" : "2014-06-22T01:08:52Z",
"source_update_interval" : 900000,
"source_setting_update_strategy" : "CYCLICAL",
"source_setting_index_strategy" : "DEFAULT",
"source_title" : "CNN",
"permalink" : "http://www.cnn.com/2014/10/15/health/texas-ebola-outbreak/index.html",
"canonical" : "http://www.cnn.com/2014/10/15/health/texas-ebola-outbreak/index.html",
"main" : "<p>Full HTML of the content</p>",
"main_length" : 31,
"main_checksum" : "I7QyvW\_g9AjGg3vWjmcxwo7wjXs",
"main_format" : "HTML",
"summary_text" : "Another nurse who contracted Ebola after caring for a man who died of the virus
  was on a flight from Cleveland to Dallas.",
"title" : "CDC: Nurse with Ebola should not have traveled",
"publisher" : "CNN",
"section" : "Technology",
"description" : "Another nurse who contracted Ebola after caring for a man who died of the virus
 was on a flight from Cleveland to Dallas.",
"tags" : [ "nurse", "outbreak", "ebola" ],
"published" : "2014-06-22T01:08:52Z",
"author_name" : "Holly Yan",
"author_link" : "https://twitter.com/HollyYanCNN",
"lang" : "en"
}
```

B.1 Step 1: Pre-filtering

For the first step, preliminary filtering of the incoming stream of content from Spinn3r, I use the following rules:

- If the source is twitter, then a post is considered earnings related if the content contains the ticker of a Russel3000 company preceded by a $-sign.
- If the source is not twitter, then a post is considered earnings related, if the content contains 3 distinct keywords from a keyword list and either the name of a Russel3000 company or its ticker.

However, for the sample used in this study, twitter posts where excluded from the classifier and sample construction. The keyword list can be found in Exhibit 6. These are selected based on the results of a Naive Bayesian Classifier trained on a sample of blog posts from 5 blogs with different topics. Two blogs contain investment and earnings related news: MoneyBeat (http://blogs.wsj.com/moneybeat/) and posts from the SeekingAlpha site (http://seekingalpha.com/). The three others are chosen as representative for non-earnings related posts: politico.com, espn.com, and entertainment-online.com. I use the sites' main rss feed and pulled news content every 20 minutes from the 3rd of January to 1st of May, 2015. The resulting sample consists of 12,930 blog posts, with 10,401 from MoneyBeat and SeekingAlpha. I label those as business related and train the classifier to find those keywords that are most predictive for a text including those terms to be a business related post. I remove boilerplate sentences such as "receive our newsletter via email" before creating the term frequency matrix, which is a matrix having a number of rows equal to the length of the vocabulary used in all 12,930 blog posts (after removing boilerplate and stopwords, like "has", "she", and "and") and 12,930 columns, one for each post. Each row contains the frequency of one term in the vocabulary appearing in the respective posts.

Exhibit 6: Keyword list used for first pass on raw Spinn3r data*

*a keyword is considered matched if it is preceded by whitespace and followed by any non-number character.

lower	futur	million
number	stock	compar
operat	business	expect
market	invest	increas
compani	growth	company
sales	valu	manag
investor	financi	quarter
revenu	earnings	risk
cost	share price	result
posit	report	trade
capital	year	

The classifier was trained with the following pipeline: First, a CountVectorizer computes the term frequency matrix (a matrix of all terms occurring in all posts as rows, each post being a

column and each cell being the frequency of each term appearing in a post). Then, the Term-frequency/inverse document frequency matrix is then considered. Next, the number of features is reduced to 1,000, using a feature selector that selects the 1,000 features with the highest k-score. Last a Naive Bayes Classifier was trained. I chose those hyper-parameters that produced the highest classification accuracy based on a 5-fold cross-validation. The code for the whole procedure is available on request. Posts are classified based on the list of keywords and are stored in a database for further processing in a second stage.

B.2 Step 2: Classifying news as earnings related

In the second stage, the saved raw data from the initial stream is preprocessed (removing duplicates that where crawled from multiple sources) and then filtered for earnings related news. Next, a classifier was first chosen, then trained; based on a hand-labeled set of 26,250 news posts randomly sampled from all news posts in a [-4:+4 day] window from 46 randomly sampled earnings announcements (All code can be found in the github repository). The hand-labeling consisted of multiple steps with multiple labels, but the relevant one for the purposes of this paper are "EA" for earnings announcement related, "MA" for market activity summary, which denotes posts giving a summary of the the daily market activity", and all other labels with are summed up to "other". Apart from the labels, all data fields provided by Spinn3r can potentially be used. The three data fields that seemed most informative about the content of a news posts are its content, its title (if it has one), and its url. Accordingly, I constructed features (variables). From the title, I created features that count the occurrence of words in Exhibit 7 in the title of a post. Similarly, I split a post's url into separate terms at occurrences of the characters ("/", ".", "-", "_") and counted the occurrence of terms in Exhibit 8. The bottom terms being sites frequently publishing earnings news.

Exhibit 7: Title features

```
['earnings', 'ticker', 'quarterly', 'revenue', 'report', 'eps', 'profitable',
'q1', 'q2', 'q3', 'q4', 'posts', 'estimates', 'results', 'fy14', 'fy15', 'fy16',
'reveals', 'releases', 'guidance', 'revenues', 'loss', 'profits', 'quarter',
'annual', 'continues', 'delivers', 'fy', 'profit', 'expectations',
'sales', 'forecast', 'forecasts', 'posts',
'stocks', 'nasdaq', 'beats', 'gains', 'market', 'futures',
'dow', 'lower', 'record', 'quarterly', 'mixed', 'movers',
'nyse', 'wall', 'higher', 'looms', 'stock', 'busy', 'dollar',
'quarter', 'trading']
```

Both term lists are based on a simple chi-squared test commonly used in natural language processing (Manning, Schütze et al., 1999). The frequency of all terms are independently compared between posts labeled as 'earnings related' and 'other', the chi-square statistic of significant differences in frequency computed, and the words with the highest frequency were retained from that list (unless they were clearly over-fitting. For example, the word 'cloud' appeared at the top

of the list, because it was a common theme in the sample earnings announcements, but I choose to not retain it as to not bias the classifier towards tech companies).

Exhibit 8: URL features

```
['revenue', 'revenues', 'loss', 'profitable', 'profits', 'quarter',
'earnings', 'annual', 'estimates', 'continues', 'delivers', 'fy',
'profit', 'expectations', 'sales', 'forecast', 'forecasts', 'posts',
'guidance',
'stocks', 'wall', 'street', 'market', 'markets', 'futures',
'movers', 'fed', 'nasdaq', 'after', 'stock', 'mixed',
'morning', 'rise', 'dow', 'higher', 'record', 'investors',
'beats', 'early', 'shares',
'gains', 'lower' 'dollar',
'tops', 'looms', 'bell', 'split', 'trading',
'lightreading', 'tickerreport', 'yahoo', 'zacks', 'trefis',
'ecommercetactics', 'barrons', 'watchlistnews', 'americasmarkets',
'cnbc', 'us', 'itnews', 'online', 'facts', 'cnn', 'wsj',
'techinvestornews', 'nytimes', 'reuters', 'fool', 'northjersey',
'omaha', 'access-wallstreet', 'electronista', 'fortune', 'sleekmoney',
'lulegacy', 'marketrealist', 'ph', 'newsdaily', 'usnews', 'feeds',
'americanbankingnews', 'net', 'financialeveryday', 'capitalbay',
'wkrb13', 'gearor', 'americanpublicmedia', 'breakingviews',
'theartofservice', 'blogspot', 'nasdaq', 'technewstube',
'canadianbusiness', 'waaytv', 'finance', 'abqjournal', 'usatoday',
'rttnews', 'marketwatch', 'benzinga', 'thestockmarketwatch',
'thestreet', 'proactiveinvestors', 'thefly', 'streetinsider',
'tradingfloor', 'foxbusiness', 'fxstreet',
'findata', 'financialtides']
```

From the text of the news posts, I first create some low-level text features: The number of firms mentioned (based on a list of Russel 3000 firm names), the highest count of occurrences of a single firm name, the sum of all firm mentions, the difference in days between the publication of this post and the earnings announcement, the number of times the candidate firm name is mentioned, the length of the news post in characters, the count ratio, that is the ratio of how many times the candidate firm name appeared compared to the name with the highest count, and the number of tickers mentioned in a news post.

I proceed to construct a term-frequency matrix from the text to complement the feature set. I tokenize the text into terms using the nltk package in Python and retain the top 200 tokens and the bottom 200 hundred tokens of a news post. As the topic of a post must be apparent from the beginning of a post and the bottom of a post usually has characteristic boilerplate, especially if it is non-financial news. Removing the middle of large posts, did not significantly reduce classification accuracy and sped up classification significantly. Then, I remove all one character tokens. I used the nltk part-of-speech tagger to identify and keep nouns, verbs and adjectives only. Part of speech tagging allows to differentiate terms, such as 'run' (noun) and 'to run' (verb), so that they appear as different terms in the matrix. I keep only terms with more than 2 and less than 25 characters (they nearly always represent garbled words or plain text urls). I also remove terms that contain special characters, such as '?', and digits. Finally, words are stemmed (endings such as -ing are removed, so that run and running are recognized as one word) using the Snowball Stemmer in nltk. The remaining terms are used to build a term frequency matrix, that is a matrix where each

row is a news post, each column is a word, and the cell contents are the frequency of occurrence of the term in that post.

These features, constructed for all 26,250 hand-labeled posts, are the raw data on which the appropriate classifier was chosen and trained. To decide on the most suitable classifier, I followed a standard procedure (Witten, Frank, Hall, and Pal, 2016) and ran a horse race between various possible pre-processing steps and three types of estimators most commonly used for text classification (Naive Bayes, Logistic Regression, and Support Vector Machines) with varying hyper-parameters. The possible pre-processing steps I considered were: equal-weighting or down-weighting the title and url features, only considering the 500, 1000, or 1500 most discriminating terms (based on chi-squared values) of the term-frequency matrix, considering or not considering N-grams consisting of up to three words, removing terms that appear in less than 2% of the sample posts or removing words that appear in less than 10 sample posts. Terms that appear in more than 80% of all posts are always removed because by definition they won't help much in discriminating themes and removing them reduces over-fitting. Taken together with the three choices of estimators this amounts to 2 x 3 x 2 x 2 x 3 x "Combination of hyper-parameter choices" candidate classifier pipelines that were compared to each other. To compared them I employed 5-fold cross-validation on a test sample. First, 20% of the posts were taken out and not used for the horse race. The 80% train sample was then randomly split into 5 parts (while making sure the proportion of earnings related and other posts is roughly the same per part). Then 4 parts are used to estimate the model and the predictive ability of the model is assessed by applying the model to predict labels ("earnings-related", "other", "market summary") on the left-out 5th piece of news posts. The predictive accuracy on this hold-out piece is saved. Then procedure is repeated for another combination of four pieces until every piece was a hold-out piece once. The average predictive accuracy across all five pieces is stored and used to choose the best classifier pipeline. The pipeline with the best score was the one employing logistic regression with balanced classes (accounting for the fact that earnings news are rare compared to the number of other news topics) and L1 penalty criterion. This also fits the conclusions and recommendations by Gentzkow, Kelly, and Taddy (2017). The best pre-processing steps were the combination of equally weighting all feature sources, no consideration of N-grams, removing terms that appear in less than 2% of the news posts, keeping the 1500 most discriminating terms. The test accuracy was 0.928. The actual performance of the so chosen classifier was assessed based on a test sample made up of the 20% of the hand-labeled sample that was not used for the horse race. The precision and recall on that test set are presented in Table 9.

While not perfect, the classifier performs reasonably well (in fact better than similar classifiers used in the literature (e.g., Li, 2010). Upon checking, most of the misclassification are news about analyst recommendations, reprints of earnings releases, and in some cases earnings news to banks or related companies. All three introduce noise into the dataset. The first two are accounted for

Table 9: Precision and recall of final classifier on test sample

Label	Precision	Recall	F1-Score	Support
Other	0.97	0.94	0.96	3630
EA	0.71	0.88	0.79	427
MA	0.86	0.87	0.86	581
Average	0.93	0.93	0.93	4638

[a] Precision is the fraction of posts labeled correctly as "x" of all posts that were labeled "x". Recall (also known as sensitivity) is the fraction of all true "x" posts that were labeled "x". The F1-Score is the harmonic mean of precision and recall.

in the audience model by including earnings announcement specific fixed effects. The third cause is rare is and therefore considered a negligible source of noise.

The final classifier is applied separately for each earnings announcement by first searching the database for all news in a [0; +4 day] window around an earnings announcement and which contain either the name or ticker of the announcing firm. Then the classifier is applied to that subset and the earnings related news and market summary posts are retained. From this the final sample is merged.