Text Mining in FinTech

Indian School of Business | FinTech | 2018

Sanjiv Ranjan Das, Professor of Finance and Data Science
Santa Clara University

Text and Context: Language Analytics for Finance

Text as Data

1. Big Text: there is more textual data than numerical data.
2. Text is versatile. Nuances and behavioral expressions that are not conveyed with numbers.
3. Text contains emotive content. Sentiment analysis. Admati-Pfleiderer 2001; DeMarzo et al 2003; Antweiler-Frank 2004, 2005; Das-Chen 2007; Tetlock 2007; Tetlock et al (2008); Mitra et al 2008; Leinweber-Sisk 2010.
4. Text contains opinions and connections. Das et al 2005; Das and Sisk 2005; Godes et al 2005; Li 2006; Hochberg et al 2007.
5. Numbers aggregate; text disaggregates.

Anecdotal ....

- In a talk at the 17th ACM Conference on Information Knowledge and Management (CIKM '08), Google's director of research Peter Norvig stated his unequivocal preference for data over algorithms—"data is more agile than code." Yet, it is well-understood that too much data can lead to overfitting so that an algorithm becomes mostly useless out-of-sample.
- Chris Anderson: "Data is the New Theory."

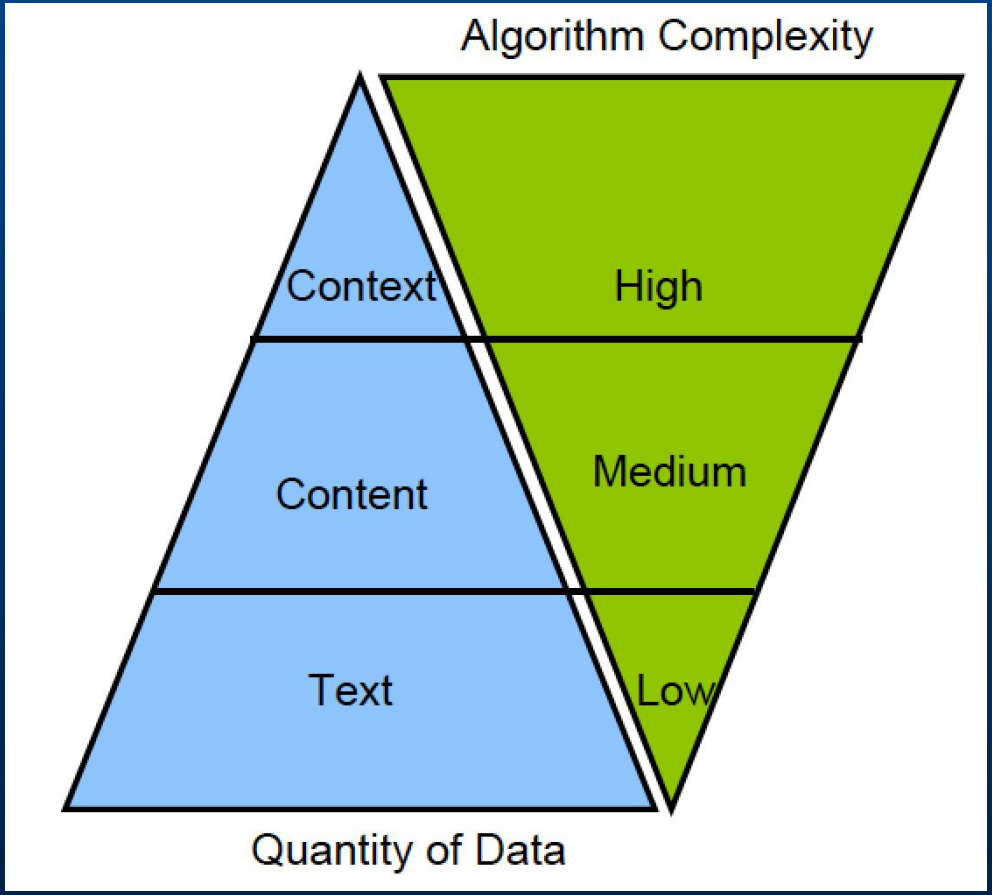
Definition: Text Mining

- Text mining is the large-scale, automated processing of plain text language in digital form to extract data that is converted into useful quantitative or qualitative information.
- Text mining is automated on big data that is not amenable to human processing within reasonable time frames. It entails extracting data that is converted into information of many types.
- Simple: Text mining may be simple as in key word searches and counts.
- Complicated: It may require language parsing and complex rules for information extraction.
- Structured text, such as the information in forms and some kinds of web pages.
- Unstructured text is a much harder endeavor.
- Text mining is also aimed at unearthing unseen relationships in unstructured text as in meta analyses of research papers, see Van Noorden 2012.

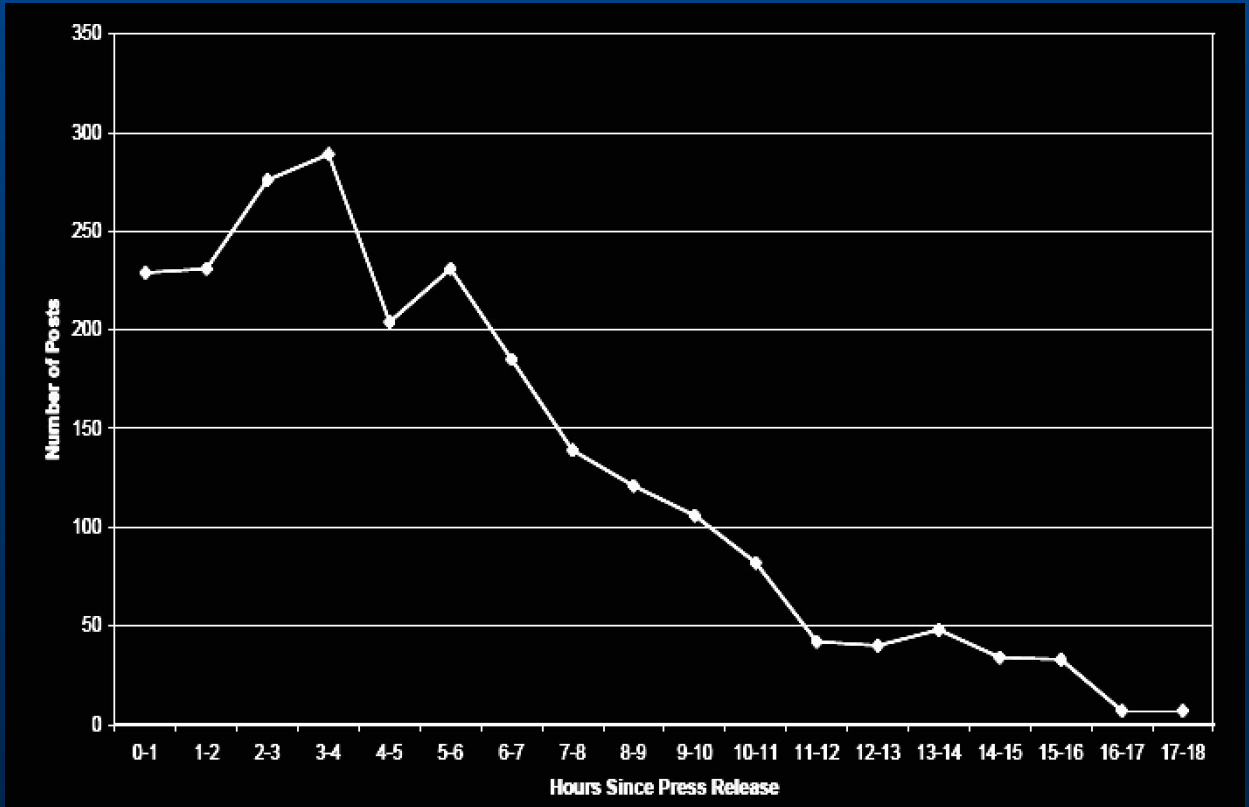
Definition: News Analytics

Wikipedia defines it as - "... the measurement of the various qualitative and quantitative attributes of textual (unstructured data) news stories. Some of these attributes are: sentiment, relevance, and novelty. Expressing news stories as numbers permits the manipulation of everyday information in a mathematical and statistical way. News analytics are used in financial modeling, particularly in quantitative and algorithmic trading. Further, news analytics can be used to plot and characterize firm behaviors over time and thus yield important strategic insights about rival firms. News analytics are usually derived through automated text analysis and applied to digital texts using elements from natural language processing and machine learning such as latent semantic analysis, support vector machines, 'bag of words', among other techniques."

Data and Algorithms

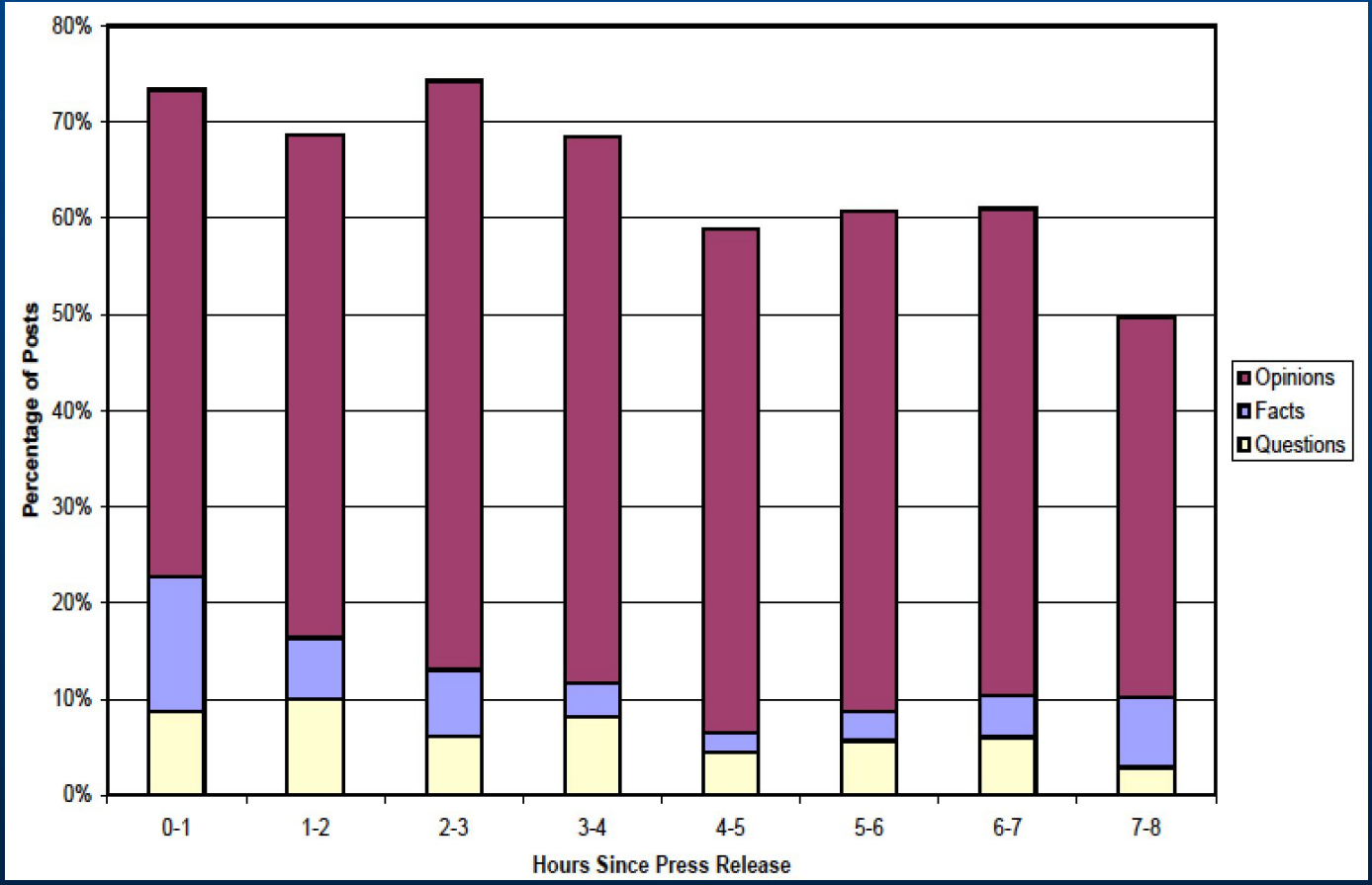


Extraction and Analysis

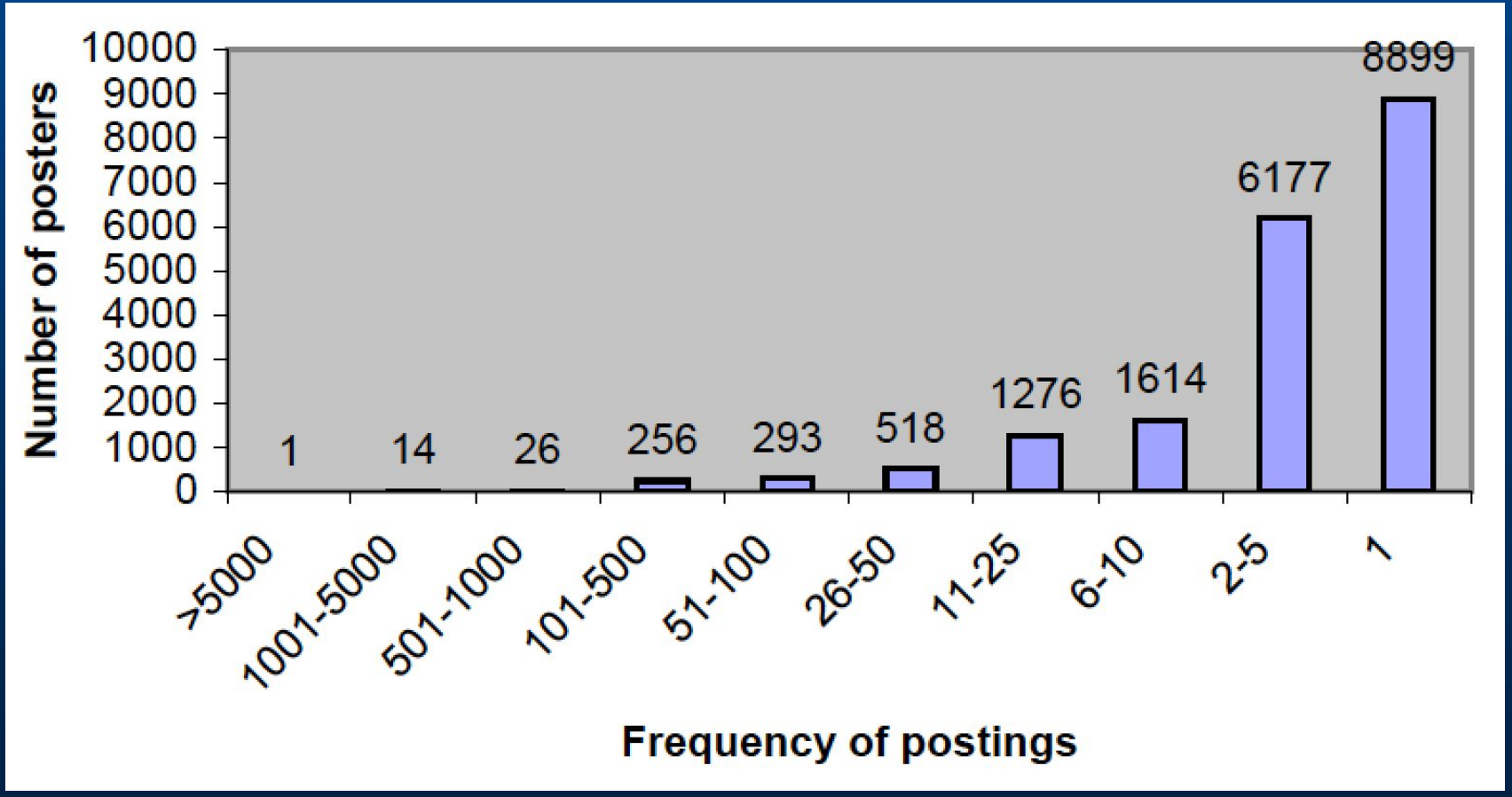


Das, Martinez-Jerez, Tufano (FM 2005)

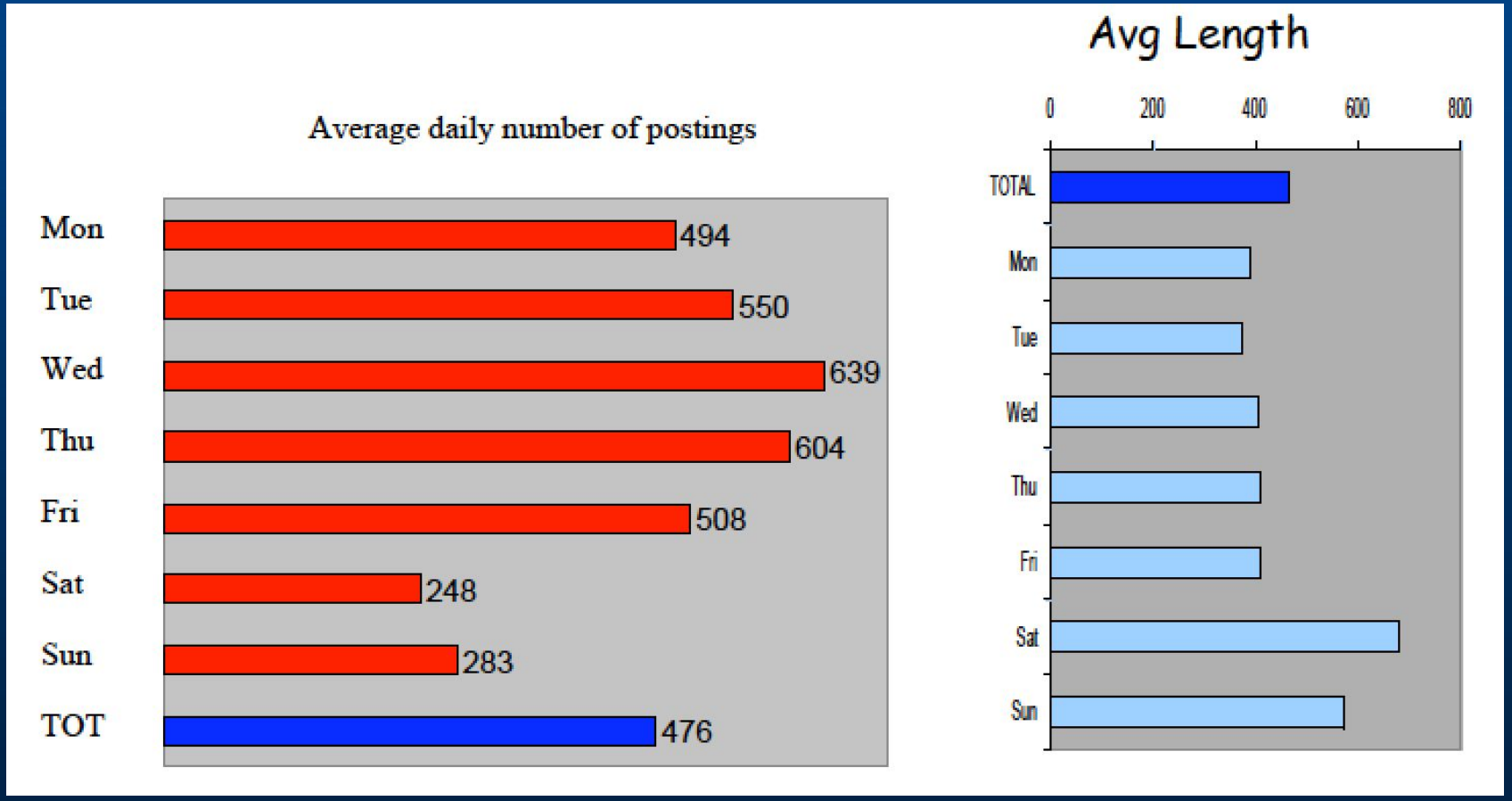
Extraction and Analysis



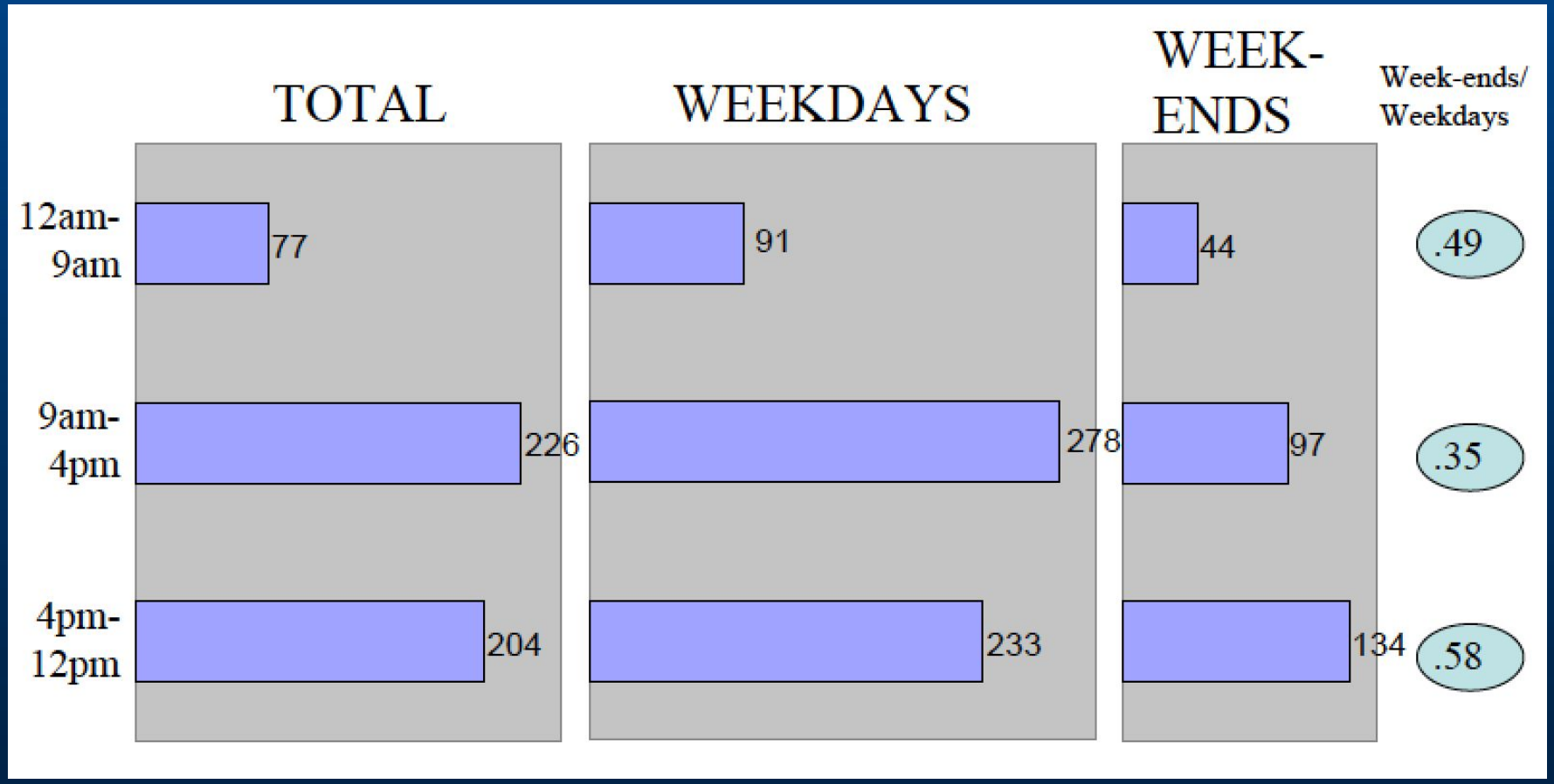
Extraction and Analysis



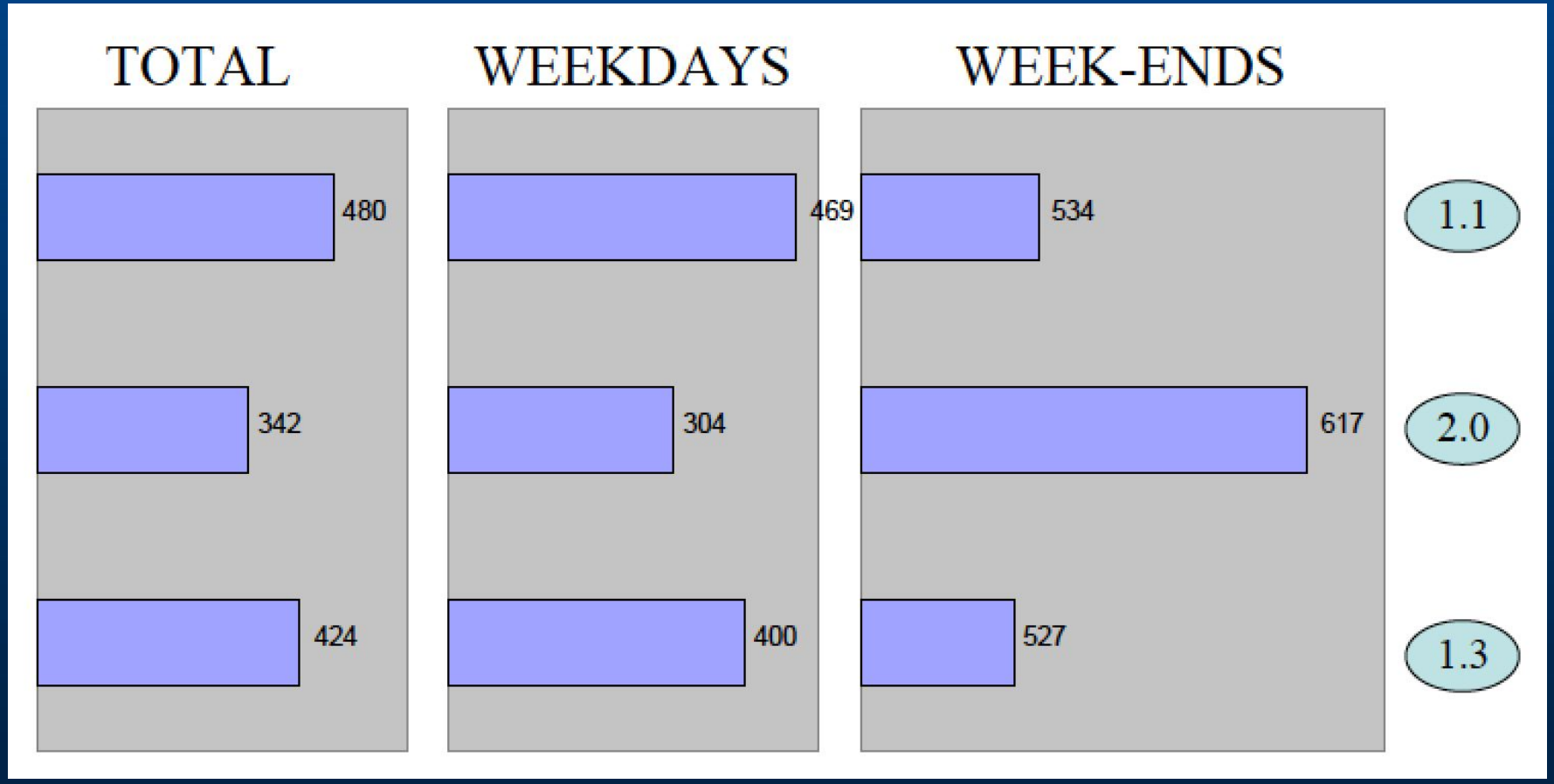
Extraction and Analysis



Extraction and Analysis



Extraction and Analysis



Dictionaries

- Webster's defines a "dictionary" as "a reference source in print or electronic form containing words usually alphabetically arranged along with information about their forms, pronunciations, functions, etymologies, meanings, and syntactical and idiomatic uses."
- The Harvard General Inquirer (`http://www.wjh.harvard.edu/ inquirer/`)
- Standard Dictionaries: `dictionary.com`, and `www.merriam-webster.com`.
- Computer dictionary: `http://www.hyperdictionary.com/computer` that contains about 14,000 computer related words, such as "byte" or "hyperlink".
- Math dictionary, such as `http://www.amathsdictionaryforkids.com/dictionary.html`.
- Medical dictionary, see `http://www.hyperdictionary.com/medical`.

Dictionaries

- Internet lingo dictionaries may be used to complement standard dictionaries with words that are not usually found in standard language, for example, see `http://www.netlingo.com/dictionary/all.php` for words such as `2BZ4UQT` which stands for "too busy for you cutey" (LOL).
- Associative dictionaries are also useful when trying to find context, as the word may be related to a concept, identified using a dictionary such as `http://www.visuwords.com/`. This dictionary doubles up as a thesaurus, as it provides alternative words and phrases that mean the same thing, and also related concepts.
- Value dictionaries deal with values and may be useful when only affect (positive or negative) is insufficient for scoring text. The Lasswell Value Dictionary `http://www.wjh.harvard.edu/~inquirer/lasswell.htm` may be used to score the loading of text on the eight basic value categories: Wealth, Power, Respect, Rectitude, Skill, Enlightenment, Affection, and Well being.

Lexicons

- A "lexicon" is defined by Webster's as "a book containing an alphabetical arrangement of the words in a language and their definitions; the vocabulary of a language, an individual speaker or group of speakers, or a subject; the total stock of morphemes in a language." This suggests it is not that different from a dictionary.
- A "morpheme" is defined as "a word or a part of a word that has a meaning and that contains no smaller part that has a meaning."
- In the text analytics realm, we will take a lexicon to be a smaller, special purpose dictionary, containing words that are relevant to the domain of interest.
- The benefit of a lexicon is that it enables focusing only on words that are relevant to the analytics and discards words that are not.
- Another benefit is that since it is a smaller dictionary, the computational effort required by text analytics algorithms is drastically reduced.

Constructing a Lexicon

- By hand. This is an effective technique and the simplest. It calls for a human reader who scans a representative sample of text documents and culls important words that lend interpretive meaning.
- Examine the term document matrix for most frequent words, and pick the ones that have high connotation for the classification task at hand.
- Use pre-classified documents in a text corpus. We analyze the separate groups of documents to find words whose difference in frequency between groups is highest. Such words are likely to be better in discriminating between groups.

Lexicons as Word Lists

- Das and Chen (2007) constructed a lexicon of about 375 words that are useful in parsing sentiment from stock message boards. This lexicon also introduced the notion of "negation tagging" into the literature.
- Loughran and McDonald (2011):
  - Taking a sample of 50,115 firm-year 10-Ks from 1994 to 2008, they found that almost three-fourths of the words identified as negative by the Harvard Inquirer dictionary are not typically negative words in a financial context.
  - Therefore, they specifically created separate lists of words by the following attributes of words: negative, positive, uncertainty, litigious, strong modal, and weak modal. Modal words are based on [?]'s categories of strong and weak modal words. These word lists may be downloaded from `http://www3.nd.edu/~mcdonald/Word_Lists.html`.

Text Summarization

A document D is comprised of m sentences $s_i, i = 1, 2, ..., m$, where each s_i is a set of words. We compute the pairwise overlap between sentences using the **Jaccard** similarity index:

$$J_{ij} = J(s_i, s_j) = \frac{|s_i \cap s_j|}{|s_i \cup s_j|} = J_{ji}$$

The overlap is the ratio of the size of the intersect of the two word sets in sentences s_i and s_j, divided by the size of the union of the two sets. The similarity score of each sentence is computed as the row sums of the Jaccard similarity matrix.

$$\mathcal{S}_i = \sum_{j=1}^{m} J_{ij}$$

Once the row sums are obtained, they are sorted and the summary is the first n sentences based on the $\mathcal{S}_i$ values.

Text Classification

- Machine classification is, from a layman's point of view, nothing but learning by example. In new-fangled modern parlance, it is a technique in the field of "machine learning".
- Learning by machines falls into two categories, supervised and unsupervised. When a number of explanatory X variables are used to determine some outcome Y, and we train an algorithm to do this, we are performing supervised (machine) learning. The outcome Y may be a dependent variable (for example, the left hand side in a linear regression), or a classification (i.e., discrete outcome).
- When we only have X variables and no separate outcome variable Y, we perform unsupervised learning. For example, cluster analysis produces groupings based on the X variables of various entities, and is a common example.

Bayes Classifier

Bayes classification extends the Document-Term model with a document-term-classification model. These are the three entities in the model and we denote them as (d, t, c). Assume that there are D documents to classify into C categories, and we employ a dictionary/lexicon (as the case may be) of T terms or words. Hence we have $d_i, i = 1, ..., D$, and $t_j, j = 1, ..., T$. And correspondingly the categories for classification are $c_k, k = 1, ..., C$.

Bayes Classifier

Suppose we are given a text corpus of stock market related documents (tweets for example), and wish to classify them into bullish (c_1), neutral (c_2), or bearish (c_3), where $C = 3$. We first need to train the Bayes classifier using a training data set, with pre-classified documents, numbering D. For each term t in the lexicon, we can compute how likely it is to appear in documents in each class c_k. Therefore, for each class, there is a T-sided dice with each face representing a term and having a probability of coming up. These dice are the prior probabilities of seeing a word for each class of document. We denote these probabilities succinctly as $p(t|c)$. For example in a bearish document, if the word "sell" comprises 10% of the words that appear, then $p(t = \text{sell}|c = \text{bearish}) = 0.10$.

Bayes Classifier

In order to ensure that just because a word does not appear in a class, it has a non-zero probability we compute the probabilities as follows:

$$p(t|c) = \frac{n(t|c) + 1}{n(c) + T}$$

where $n(t|c)$ is the number of times word t appears in category c, and $n(c) = \sum_t n(t|c)$ is the total number of words in the training data in class c. Note that if there are no words in the class c, then each term t has probability $1/T$.

Bayes Classifier

A document d_i is a collection or set of words t_j. The probability of seeing a given document in each category is given by the following *multinomial* probability:

$$p(d|c) = \frac{n(d)!}{n(t_1|d)! \cdot n(t_2|d)! \cdots n(t_T|d)!} \times p(t_1|c) \cdot p(t_2|c) \cdots p(t_T|c)$$

where $n(d)$ is the number of words in the document, and $n(t_j|d)$ is the number of occurrences of word t_j in the same document d. These $p(d|c)$ are the prior probabilities in the Bayes classifier, computed from all documents in the training data. The posterior probabilities are computed for each document in the test data as follows:

$$p(c|d) = \frac{p(d|c)p(c)}{\sum_k \ p(d|c_k)p(c_k)}, \forall k = 1, \ldots, C$$

Note that we get C posterior probabilities for document d, and assign the document to class $\max_k c_k$, i.e., the class with the highest posterior probability for the given document.

Word Count Classifiers

- Given a lexicon of selected words, one may sign the words as positive or negative, and then do a simple word count to compute net sentiment or mood of text. By establishing appropriate cut offs, one can determine the classification of text into optimistic, neutral, or pessimistic. These cut offs are determined using the training and testing data sets.
- Word count classifiers may be enhanced by focusing on "emphasis words" such as adjectives and adverbs, especially when classifying emotive content. One approach used in Das and Chen (2007) is to identify all adjectives and adverbs in the text and then only consider words that are within ± 3 words before and after the adjective or adverb. This extracts the most emphatic parts of the text only, and then mood scores it.

Fisher's Discriminant

- Fisher's discriminant is simply the ratio of the variation of a given word across groups to the variation within group.
- More formally, Fisher's discriminant score $F(w)$ for word w is

$$F(w) = \frac{\frac{1}{K}\sum_{j=1}^{K}(\bar{w}_j - \bar{w}_0)^2}{\frac{1}{K}\sum_{j=1}^{K}\sigma_j^2}$$

  where K is the number of categories and $\bar{w}_j$ is the mean occurrence of the word w in each text in category j, and $\bar{w}_0$ is the mean occurrence across all categories. And σ_j^2 is the variance of the word occurrence in category j. This is just one way in which Fisher's discriminant may be calculated, and there are other variations on the theme.
- We may compute $F(w)$ for each word w, and then use it to weight the word counts of each text, thereby giving greater credence to words that are better discriminants.

Vector-Distance Classifier

Suppose we have 500 documents in each of two categories, bullish and bearish. These 1,000 documents may all be placed as points in n-dimensional space. It is more than likely that the points in each category will lie closer to each other than to the points in the other category. Now, if we wish to classify a new document, with vector D_i, the obvious idea is to look at which cluster it is closest to, or which point in either cluster it is closest to. The closeness between two documents i and j is determined easily by the well known metric of cosine distance, i.e.,

$$1 - \cos(\theta_{ij}) = 1 - \frac{D_i^\top D_j}{||D_i|| \cdot ||D_j||}$$

where $||D_i|| = \sqrt{D_i^\top D_i}$ is the norm of the vector D_i. The cosine of the angle between the two document vectors is 1 if the two vectors are identical, and in this case the distance between them would be zero.

Metrics: Confusion Matrix

- Given K categories, the matrix is of dimension $K \times K$. By convention, the columns relate to the category assigned by the classifier algorithm and the rows refer to the actual category in which the text resides.
- If an algorithm has no classification ability, then the rows and columns of the matrix will be independent of each other. Under this null hypothesis, the statistic that is examined for rejection is as follows:

$$\chi^2[dof = (K-1)^2] = \sum_{i=1}^{K}\sum_{j=1}^{K}\frac{[O(i,j) - E(i,j)]^2}{E(i,j)}$$

  where $O(i,j)$ are the actual numbers observed in the confusion matrix, and $E(i,j)$ are the expected numbers, assuming no classification ability under the null hypothesis.
- If $M(i)$ represents the total across row i of the confusion matrix, and $M(j)$ the column total, then

$$E(i,j) = \frac{M(i) \times M(j)}{\sum_{i=1}^{K} M(i)} \equiv \frac{M(i) \times M(j)}{\sum_{j=1}^{K} M(j)}$$

  The degrees of freedom of the χ^2 statistic is $(K-1)^2$. This statistic is very easy to implement and may be applied to models for any K. A highly significant statistic is evidence of classification ability.

Accuracy

Algorithm accuracy over a classification scheme is the percentage of text that is correctly classified. This may be done in-sample or out-of-sample. To compute this off the confusion matrix, we calculate

$$\text{Accuracy} = \frac{\sum_{i=1}^{K} O(i,i)}{\sum_{j=1}^{K} M(j)} = \frac{\sum_{i=1}^{K} O(i,i)}{\sum_{i=1}^{K} M(i)}$$

We should hope that this is at least greater than $1/K$, which is the accuracy level achieved on average from random guessing.

False Positives

- The percentage of false positives is a useful metric to work with. It may be calculated as a simple count or as a weighted count (by nearness of wrong category) of false classifications divided by total classifications undertaken.
- For example, assume that in the example above, category 1 is BULLISH and category 3 is BEARISH, whereas category 2 is NEUTRAL. The false positives would arise from mis-classifying category 1 as 3 and vice-versa. We compute the false positive rate for illustration.

```
> Omatrix
    y
x     1  2  3
  1 22  3  1
  2  1 44  1
  3  0  3 25
> (Omatrix[1,3]+Omatrix[3,1])/sum(Omatrix)
[1] 0.01
```

The false positive rate is just 1%.

Sentiment Error

In a 3-way classification scheme, where category 1 is BULLISH and category 3 is BEARISH, whereas category 2 is NEUTRAL, we can compute this metric as follows.

$$\text{Sentiment Error} = 1 - \frac{M(j=1) - M(j=3)}{M(i=1) - M(i=3)}$$

In our illustrative example, we may easily calculate this metric.

```
> rsum = rowSums(Omatrix)
> csum = colSums(Omatrix)
> rsum
 1  2  3
26 46 28
> csum
 1  2  3
23 50 27
> 1 - (-3)/(-2)
[1] -0.5
```

The classified sentiment from the algorithm was $-3 = 23 - 27$, whereas it actually should have been $-2 = 26 - 28$. The percentage error in sentiment is 50%.

Disagreement

The metric uses the number of signed buys and sells in the day (based on a sentiment model) to determine how much difference of opinion there is in the market. The metric is computed as follows:

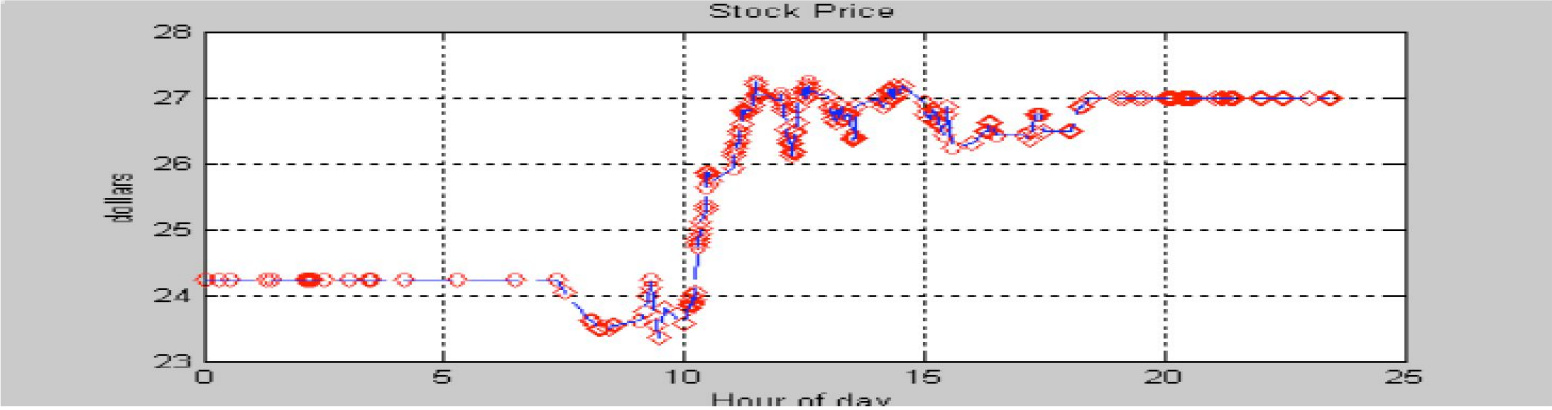
$$\text{DISAG} = \left| 1 - \left| \frac{B - S}{B + S} \right| \right|$$

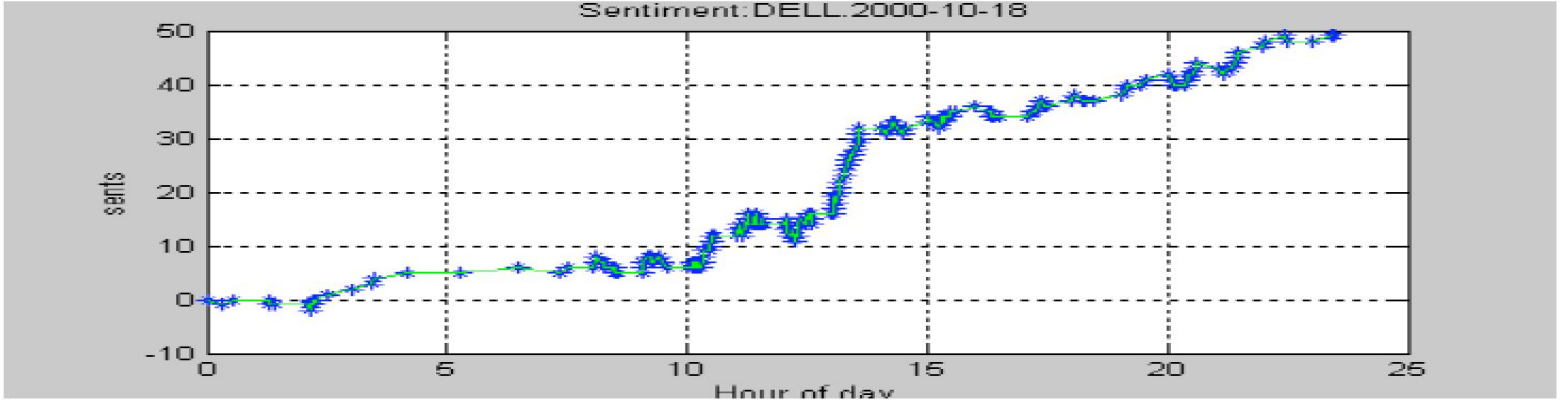
where B, S are the numbers of classified buys and sells. Note that DISAG is bounded between zero and one.
Using the true categories of buys (category 1 BULLISH) and sells (category 3 BEARISH) in the same example as before, we may compute disagreement.

```
> DISAG = abs(1-abs((26-28)/(26+28)))
> DISAG
[1] 0.962963
```

Since there is little agreement (26 buys and 28 sells), disagreement is high.

Dollars and Sents





Readability

- Gunning (1952) developed the Fog index. The index estimates the years of formal education needed to understand text on a first reading. A fog index of 12 requires the reading level of a U.S. high school senior (around 18 years old).
- The index is based on the idea that poor readability is associated with longer sentences and complex words. Complex words are those that have more than two syllables.
- The formula for the Fog index is

$$0.4 \cdot \left[\frac{\#\text{words}}{\#\text{sentences}} + 100 \cdot \left(\frac{\#\text{complex words}}{\#\text{words}} \right) \right]$$

- Standard readability metrics may not work well for financial text. Loughran-McDonald (2014) find that the Fog index is inferior to simply looking at 10-K file size.

Readability

- The Flesch Reading Ease Score is defined as

$$206.835 - 1.015\left(\frac{\#\text{words}}{\#\text{sentences}}\right) - 84.6\left(\frac{\#\text{syllables}}{\#\text{words}}\right)$$

  With a range of 90-100 easily accessible by a 11-year old, 60-70 being easy to understand for 13-15 year olds, and 0-30 for university graduates.
- The Flesch-Kincaid Grade Level is defined as

$$0.39\left(\frac{\#\text{words}}{\#\text{sentences}}\right) + 11.8\left(\frac{\#\text{syllables}}{\#\text{words}}\right) - 15.59$$

  which gives a number that corresponds to the grade level. As expected these two measures are negatively correlated.
- The Cole-Liau index does not even require a count of syllables, as follows:

$$CLI = 0.0588L - 0.296S - 15.8$$

  where L is the average number of letters per hundred words and S is the average number of sentences per hundred words.

Textual Content

- Lu, Chen, Chen, Hung, and Li (2010) categorize finance related textual content into three categories: (a) forums, blogs, and wikis; (b) news and research reports; and (c) content generated by firms.
- Extracting sentiment and other information from messages posted to stock message boards such as Yahoo!, Motley Fool, Silicon Investor, Raging Bull, etc., see Tumarkin and Whitelaw (2001), Antweiler and Frank (2004), Antweiler and Frank (2005), Das, Martinez-Jerez and Tufano (2005), Das and Chen (2007).
- Other news sources: Lexis-Nexis, Factiva, Dow Jones News, etc., see Das, Martinez-Jerez and Tufano (2005); Boudoukh, Feldman, Kogan, Richardson (2012).
- Heard on the Street column in the Wall Street Journal has been used in work by Tetlock (2007), Tetlock, Saar-Tsechansky and Macskassay (2008); see also the use of Wall Street Journal articles by Lu, Chen, Chen, Hung, and Li (2010).
- Thomson-Reuters NewsScope Sentiment Engine (RNSE) based on Infonics/Lexalytics algorithms and varied data on stocks and text from internal databases, see Leinweber and Sisk (2011). Zhang and Skiena (2010) develop a market neutral trading strategy using news media such as tweets, over 500 newspapers, Spinn3r RSS feeds, and LiveJournal.

Twitter and Facebook

- Bollen, Mao, and Zeng (2010) showed that stock direction of the Dow Jones Industrial Average can be predicted using tweets with 87.6% accuracy.
- Bar-Haim, Dinur, Feldman, Fresko and Goldstein (2011) attempt to predict stock direction using tweets by detecting and overweighting the opinion of expert investors.
- Brown (2012) looks at the correlation between tweets and the stock market via several measures.
- Logunov (2011) uses OpinionFinder to generate many measures of sentiment from tweets.
- Twitter based sentiment developed by Rao and Srivastava (2012) is found to be highly correlated with stock prices and indexes, as high as 0.88 for returns.
- Sprenger and Welpe (2010) find that tweet bullishness is associated with abnormal stock returns and tweet volume predicts trading volume.

Corporate Finance and Risk Management

- Sprenger (2011) integrates data from text classification of tweets, user voting, and a proprietary stock game to extract the bullishness of online investors; these ideas are behind the site TweetTrader.net.
- Tweets also pose interesting problems of big streaming data discussed in Pervin, Fang, Datta, and Dutta (2013).
- Data used here is from lings such as 10-Ks, etc., (Loughran and McDonald (2011); Burdick et al (2011); Bodnaruk, Loughran, and McDonald (2013); Jegadeesh and Wu (2013); Loughran and McDonald (2014)).

Predicting Markets

- Wysocki (1999) found that for the 50 top firms in message posting volume on Yahoo! Finance, message volume predicted next day abnormal stock returns. Using a broader set of firms, he also found that high message volume firms were those with inflated valuations (relative to fundamentals), high trading volume, high short seller activity (given possibly in inflated valuations), high analyst following (message posting appears to be related to news as well, correlated with a general notion of “attention” stocks), and low institutional holdings (hence broader investor discussion and interest), all intuitive outcomes.
- Bagnoli, Beneish, and Watts (1999) examined earnings “whispers", unofficial crowd-sourced forecasts of quarterly earnings from small investors, are more accurate than that of First Call analyst forecasts.
- Tumarkin and Whitelaw (2001) examined self-reported sentiment on the Raging Bull message board and found no predictive content, either of returns or volume.

Basic String Handling

```
%pylab inline
import pandas as pd
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
text = "Ask not what your country can do for you, \
but ask what you can do for your country."
```

```
#How many characters including blanks?
len(text)
```

```
83
```

Tokenize

```
#Tokenize the words, separating by spaces, periods, commas
x = text.split(" ")
print(x)
```

```
['Ask', 'not', 'what', 'your', 'country', 'can', 'do', 'for', 'you,', 'bu
t', 'ask', 'what', 'you', 'can', 'do', 'for', 'your', 'country.']
```

```
#How many words?
len(x)
```

```
18
```

Regular Expressions

```
import re
x = re.split('[ ,.]',text)
print(x)
```

```
['Ask', 'not', 'what', 'your', 'country', 'can', 'do', 'for', 'you', '',
'but', 'ask', 'what', 'you', 'can', 'do', 'for', 'your', 'country', '']
```

```
#Use a list comprehension to remove spaces
x = [j for j in x if len(j)>0]
print(x)
```

```
['Ask', 'not', 'what', 'your', 'country', 'can', 'do', 'for', 'you', 'bu
t', 'ask', 'what', 'you', 'can', 'do', 'for', 'your', 'country']
```

```
len(x)
```

```
18
```

Unique Words

```
#Unique words
y = [j.lower() for j in x]
z = unique(y)
print(z)
```

```
['ask' 'but' 'can' 'country' 'do' 'for' 'not' 'what' 'you' 'your']
```

```
len(z)
```

```
10
```

List Comprehensions

```
#Find words greater than 3 characters
[j for j in x if len(j)>3]
```

```
['what', 'your', 'country', 'what', 'your', 'country']
```

```
#Find capitalized words
[j for j in x if j.istitle()]
```

```
['Ask']
```

```
#Find words that end in t
[j for j in x if j.endswith('t')]
```

```
['not', 'what', 'but', 'what']
```

Find words containing a letter

```
#Find words that contain a
[j for j in x if "a" in set(j.lower())]
```

```
['Ask', 'what', 'can', 'ask', 'what', 'can']
```

Or, use regular expressions to help us with more complex parsing.

For example `'@[A-Za-z0-9_]+'` will return all words that:

- start with `'@'` and are followed by at least one:
- capital letter (`'A-Z'`)
- lowercase letter (`'a-z'`)
- number (`'0-9'`)
- or underscore (`'_'`)

```
#Find words that contain 'a' using RE
[j for j in x if re.search('[Aa]',j)]
```

```
['Ask', 'what', 'can', 'ask', 'what', 'can']
```

Test type of tokens

```
#Test type of tokens
print(x)
[j for j in x if j.islower()]
```

```
['Ask', 'not', 'what', 'your', 'country', 'can', 'do', 'for', 'you', 'but', 'ask', 'what', 'you', 'can', 'do', 'for',
'your', 'country']

['not',
 'what',
 'your',
 'country',
 'can',
 'do',
 'for',
 'you',
 'but',
 'ask',
 'what',
 'you',
 'can',
 'do',
 'for',
 'your',
 'country']
```

```
[j for j in x if j.isdigit()]
```

```
[]
```

```
[j for j in x if j.isalnum()]
```

```
['Ask',
 'not',
 'what',
 'your',
 'country',
 'can',
 'do',
 'for',
 'you',
 'but',
 'ask',
 'what',
 'you',
 'can',
 'do',
 'for',
 'your',
 'country']
```

String Operations

```
y = '  To be or not to be.  '
print(y.strip())
print(y.rstrip())
print(y.lower())
print(y.upper())
```

```
To be or not to be.
  To be or not to be.
  to be or not to be.
  TO BE OR NOT TO BE.
```

```
#Return the starting position of the string
print(y.find('be'))
print(y.rfind('be'))
```

```
5
18
```

```
print(y.replace('be','do'))
```

```
  To do or not to do.
```

```
y = 'Supercalifragilisticexpialidocious'
ytok = y.split('i')
print(ytok)
print('i'.join(ytok))
print(list(y))
```

```
['Supercal', 'frag', 'l', 'st', 'cexp', 'al', 'doc', 'ous']
Supercalifragilisticexpialidocious
['S', 'u', 'p', 'e', 'r', 'c', 'a', 'l', 'i', 'f', 'r', 'a', 'g', 'i', 'l', 'i', 's', 't', 'i', 'c', 'e', 'x', 'p',
'i', 'a', 'l', 'i', 'd', 'o', 'c', 'i', 'o', 'u', 's']
```

Reading in a URL

```
## Reading in a URL
import requests

url = 'http://srdas.github.io/bio-candid.html'
f = requests.get(url)
text = f.text
print(text)
```

```
<HTML>
<BODY background="http://algo.scu.edu/~sanjivdas/graphics/back2.gif">

Sanjiv Das is the William and Janice Terry Professor of Finance and
Data Science at Santa Clara University's Leavey School of Business. He
previously held faculty appointments as Associate Professor at Harvard
Business School and UC Berkeley. He holds post-graduate degrees in
Finance (M.Phil and Ph.D. from New York University), Computer Science
(M.S. from UC Berkeley), an MBA from the Indian Institute of
Management, Ahmedabad, B.Com in Accounting and Economics (University
of Bombay, Sydenham College), and is also a qualified Cost and Works
Accountant (AICWA). He is a senior editor of The Journal of Investment
Management, co-editor of The Journal of Derivatives and The Journal of
Financial Services Research, and Associate Editor of other academic
journals. Prior to being an academic, he worked in the derivatives
business in the Asia-Pacific region as a Vice-President at
Citibank. His current research interests include: machine learning,
social networks, derivatives pricing models, portfolio theory, the
modeling of default risk, systemic risk, and venture capital.  He has
published over ninety articles in academic journals, and has won
numerous awards for research and teaching. His recent book
"Derivatives: Principles and Practice" was published in May 2010
(second edition 2016).  He currently also serves as a Senior Fellow at
the FDIC Center for Financial Research.
```

```
len(text)
```

```
4113
```

```
lines = text.splitlines()
print(len(lines))
lines[3]
```

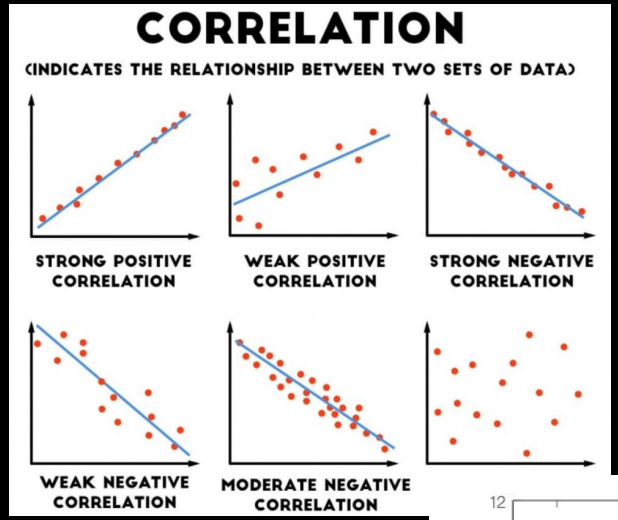
```
80

'Sanjiv Das is the William and Janice Terry Professor of Finance and'
```

Text Analysis Notebooks

- TextMining.ipynb
- TwitterAnalysis.ipynb
- IndiaNewsExtractor.ipynb

Levels of Dependence



Prediction

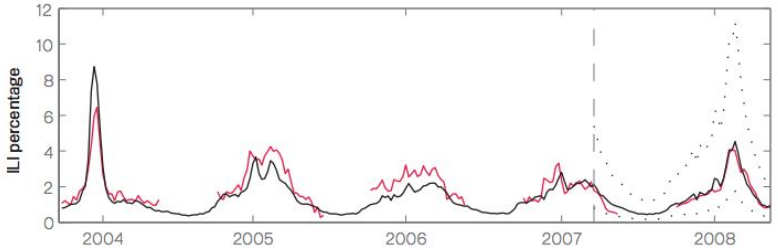
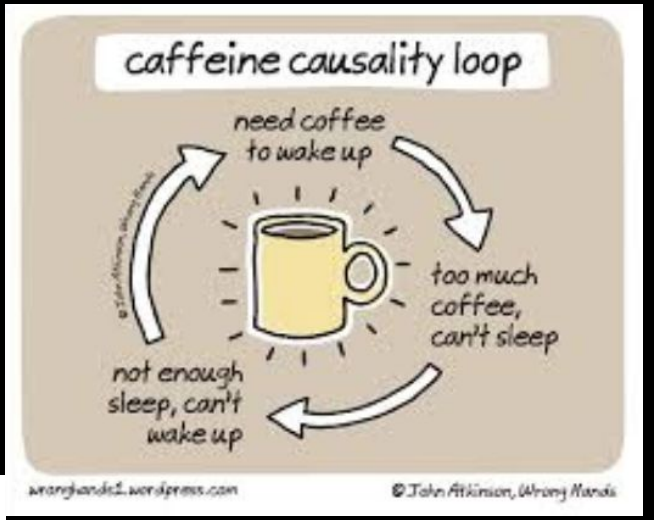


Figure 2: A comparison of model estimates for the Mid-Atlantic Region (black) against CDC-reported ILI percentages (red), including points over which the model was fit and validated. A correlation of 0.85 was obtained over 128 points from this region to which the model was fit, while a correlation of 0.96 was obtained over 42 validation points. 95% prediction intervals are indicated.



Causality

Sentiment Scoring

Loughran and McDonald JF 2011 word lists

- Fin-Neg – negative words (e.g., *loss, bankruptcy, indebtedness, felony, misstated, discontinued, expire, unable)*. N=2,349
- Fin-Pos – positive words (e.g., *beneficial, excellent, innovative*). N = 354
- Fin-Unc – uncertainty words. Note here the emphasis is more so on uncertainty than risk (e.g., ambiguity, approximate, assume, risk). N = 291
- Fin-Lit – litigious words (e.g., admission, breach, defendant, plaintiff, remand, testimony). N = 871

Negation and Stemming

- *Negation tagging*: Notice that in financial reporting it is unlikely that negative words will be negated (e.g., not terrible earnings), whereas positive words are easily qualified or compromised. Although you can easily account for simple negation, typical forms of negation are difficult to detect.
- *Stemming* does not work well for morphologically rich languages
- *Relevance* and *Novelty* : do these features matter? (“unusualness”)

t-SNE

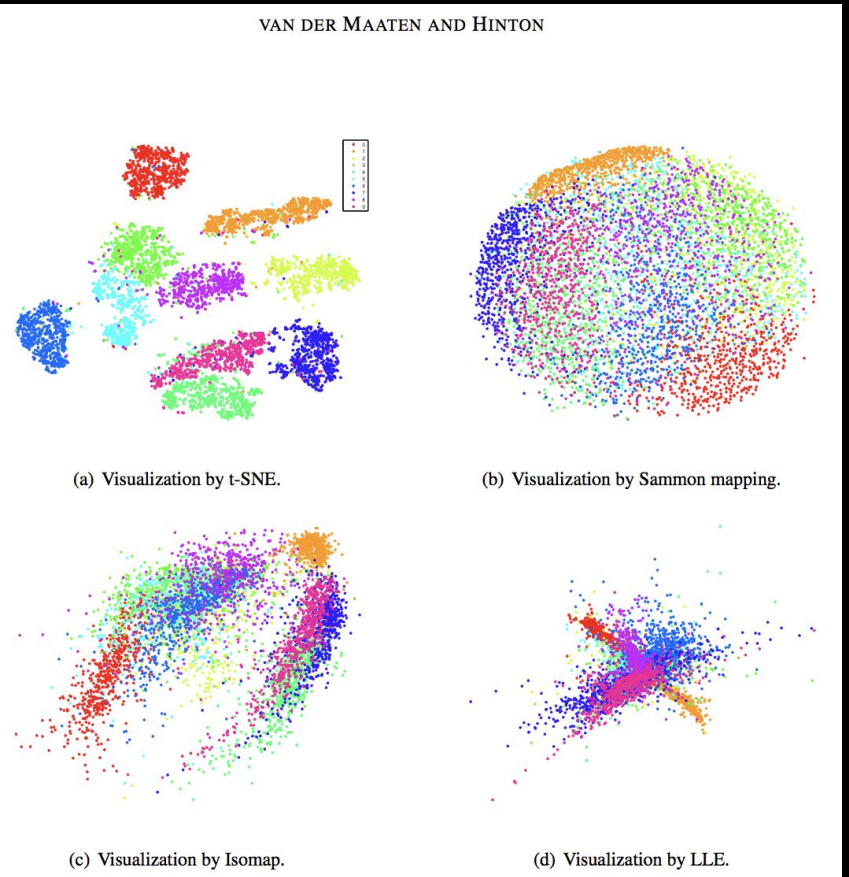
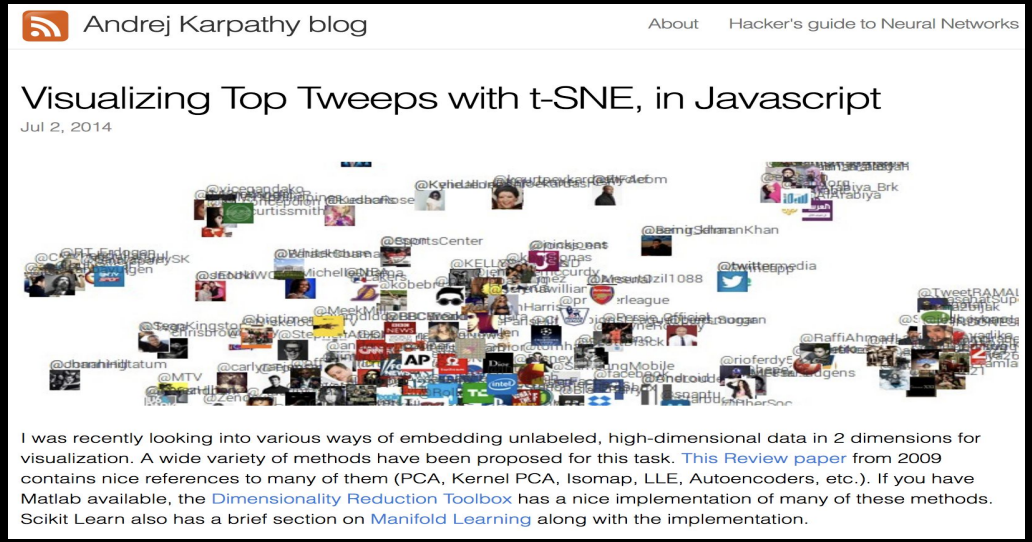


Figure 2: Visualizations of 6,000 handwritten digits from the MNIST dataset.

- t-SNE (van der Maaten and Hinton, 2008) is based on Stochastic Neighbor Embedding developed by Hinton and Rouweis (2002)
- Visualizes high-dimensional data on a two or three dimensional map
- Use word2vec to create word vectors



http://cs.stanford.edu/people/karpathy/tsnejs/

Industry Practice : Ravenpack

Sources: Dow-Jones, WSJ, Barron's, and 19000 other traditional and social media sites.

USFAST Model	Large/Mid-Cap		Small-Cap	
Statistics	**Specific Ret**	**Excess Ret**	**Specific Ret**	**Excess Ret**
Annualized Return	12.89%	10.89%	32.66%	30.13%
Annualized Vol.	3.43%	4.50%	5.38%	6.12%
Information Ratio	3.76	2.42	6.07	4.92
Avg. Portfolio Size	188	190	193	193
Max. Drawdown	2.31%	6.40%	2.81%	4.42%
Turnover	85.2%	83%	89.9%	89.5%

TABLE 1: Experimental Stats. Comparison between MSCI specific returns and excess returns for a set of statistics. Results are shown for USFAST model for Large/Mid-Cap and Small-Cap.

Source: RavenPack, MSCI, October 2017

https://www.ravenpack.com/files/research/sentiment-signals-msci-barra-risk-models/?utm_campaign=msci&utm_medium=email&utm_source=link&utm_content=&utm_term=

Performance

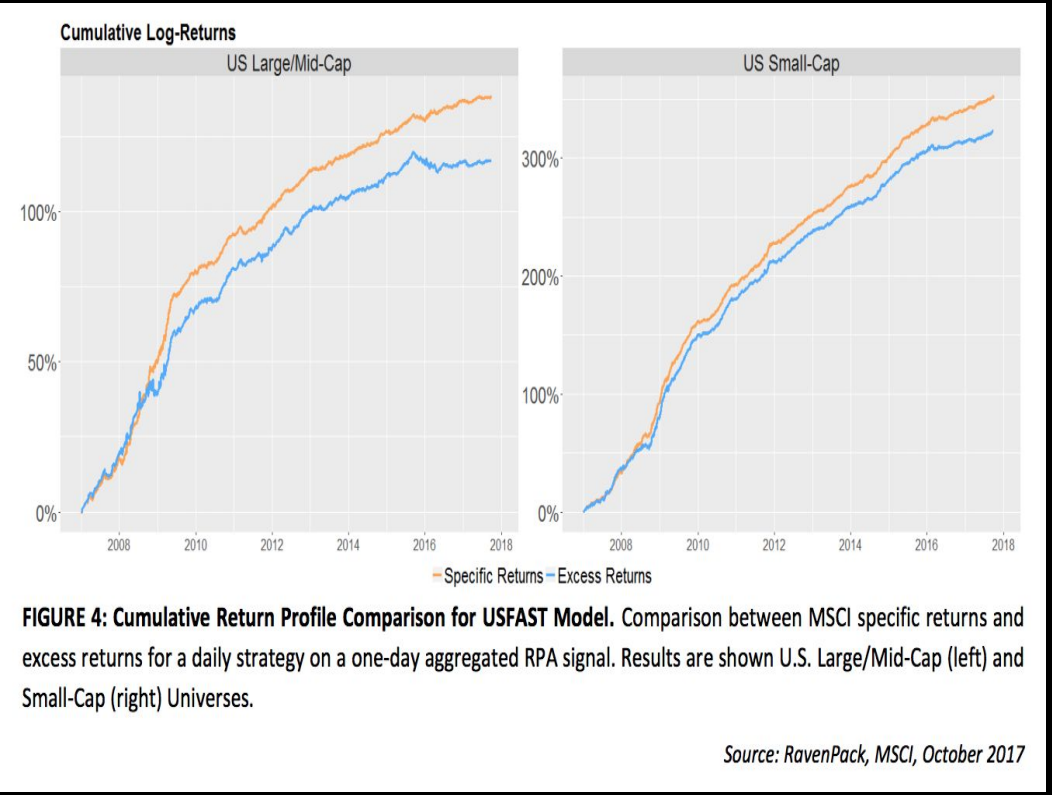


FIGURE 4: Cumulative Return Profile Comparison for USFAST Model. Comparison between MSCI specific returns and excess returns for a daily strategy on a one-day aggregated RPA signal. Results are shown U.S. Large/Mid-Cap (left) and Small-Cap (right) Universes.

Source: RavenPack, MSCI, October 2017

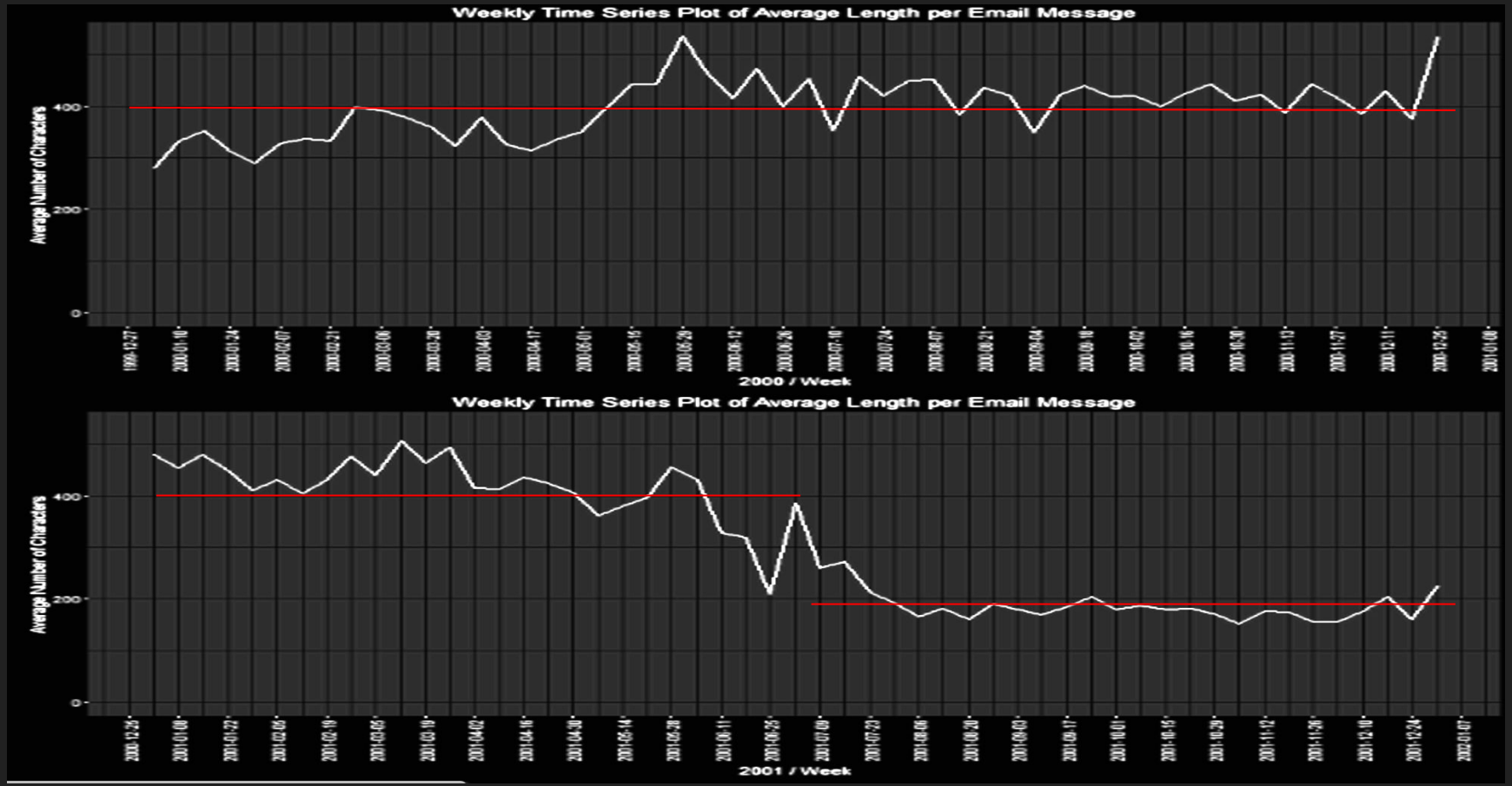
- For the U.S., Information Ratios (IR) of 3.8 and 6.1 are achieved for the Russell 1000 and the Russell 2000, respectively. We get IRs of 3.2 and 4.2 for their European equivalent.
- The predictive power of the signal is statistically significant and positive across the entire backtesting period.
- The prediction quantile analysis shows a desirable profile, providing higher returns for more extreme predictions.

RegTech

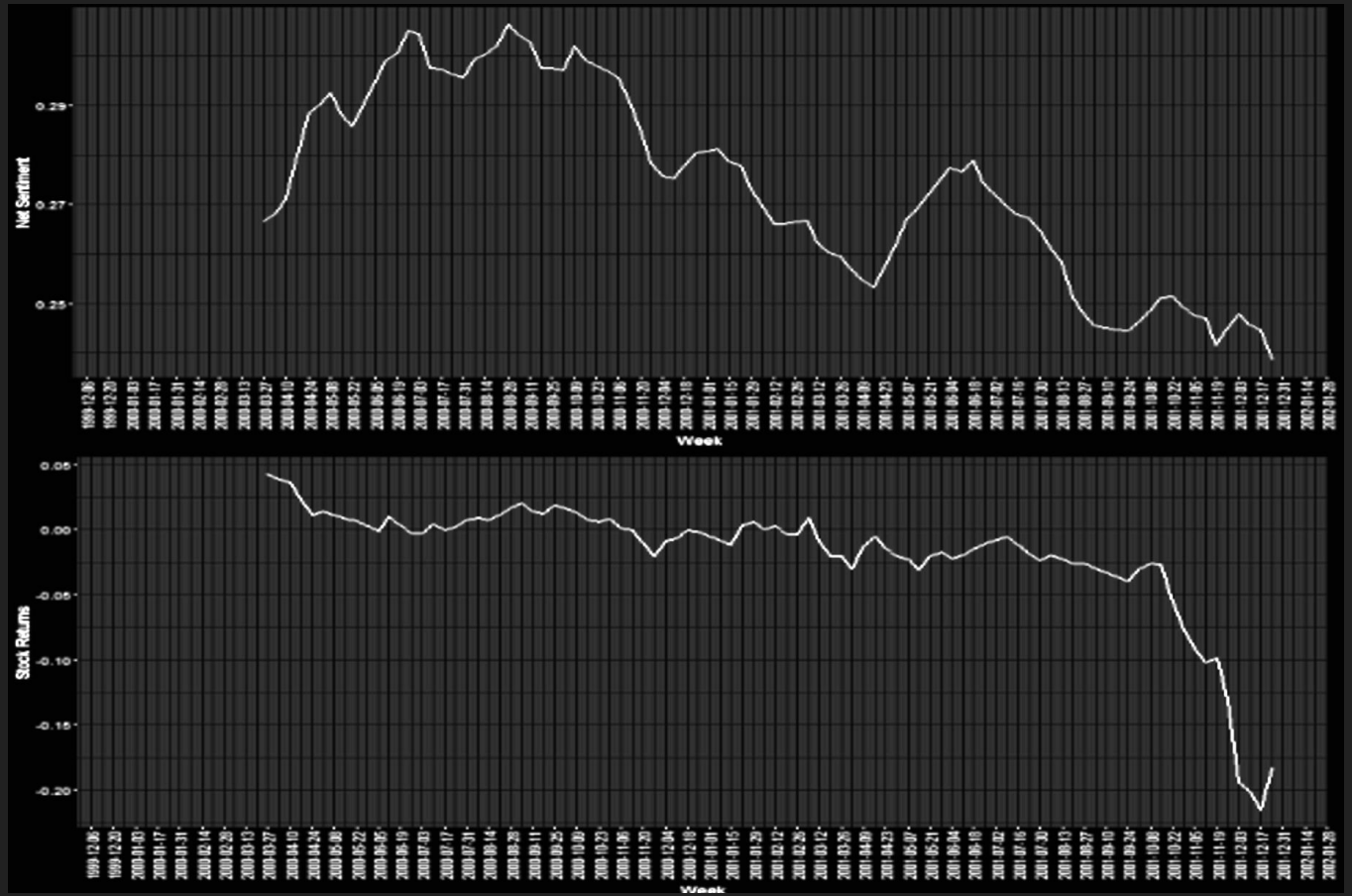
Zero-Revelation Linguistic Regulation: Detecting Risk Through Corporate Emails and News (Das, Kim, Kothari 2016)

- Financials are often delayed indicators of corporate quality.
- Internal discussion may be used as an early warning system for upcoming corporate malaise.
- Emails have the potential to predict such events.
- Software can analyze vast quantities of textual data not amenable to human processing.
- Corporate senior management may also use these analyses to better predict and manage impending crisis for their firms.
- The approach requires zero revelation of emails.

Enron: Email Length



Enron: Sentiment and Returns



Enron: Returns and Characteristics

Variable	Coefficient Estimate (*t*-statistic)			
	(1)	(2)	(3)	(4)
MA Net Sentiment $_t$	XXX***	0.575	2.330***	-1.397
	(XXX)	(0.63)	(3.14)	(-1.25)
MA Email Length $_t$		0.584***		1.046***
		(2.97)		(4.19)
MA Total Emails $_t$			-0.004	-0.131***
			(-0.10)	(-2.83)
Intercept		-0.406*	-0.671***	0.117
		(-1.93)	(-3.08)	(0.43)
Adjusted *R*-squared	XXX		0.09	0.24
Number of observations	88	88	88	88

Enron Movie (by Jim Callahan)

http://srdas.github.io/Presentations/JimCallahan_enron-sm.mov