Introduction:

This project outlines a possible design for a Natural Language Processing (NLP) based asset-pricing statistic generator. Several months ago Professors William N. Goetzmann and Robert J. Shiller gave me the opportunity to work for them as a research assistant exploring the current state of predictive computing. Presented here is a generalized outline of a promising NLP system design. This system’s most appealing features are that its algorithm is intuitive, parallizable, and modular. These features are critical in a financial forecasting system. The Directed Acyclic Graph (DAG) and partition algorithms we employ are meant to be intuitive. The idea for this algorithm was derived from “Meme-Tracking and the Dynamics of the News Cycle” (Leskovec et al). Likewise, the modular structure of the system design makes its further development easier.

System Description:

For clarity I have divided the program into four stages. Each stage is individually parallelizable. Sample data is provided to show the format of the input.

Stage 1:
(input file filled with .txt documents — dated via .xlsx metadata)
-function->
(Set of nodes V -- each node is a string extracted by a set of rules)

Stage 2:
(Set of nodes V)
-function->
(DAG(A,E))

Stage 3:
(DAG(A,E))
-function-> * DAG-Partition
(Set of narratives — where each partition of the graph is a “narrative”)

Stage 4:
Visualization / analysis of outputs of stage 3

Stage 1: We begin with a directory name, where the directory contains the text we hope to process (which must have a time dimension). This prototype system uses the input directory format exemplified by the included “WSJSample.” You will have a set of .txt files (a.k.a. strings) where each has a corresponding time. Think of this as an article, tweet, or perhaps even text generated via transcription of live videos on a social media platform at a given point in time.
This depends on the data you wish to build your system to process, but you must start with strings that have timestamps. Now you must parse through all your .txt files (parallelizable as shown). You must extract a set of all strings which you *(determine to be the most important)* and date each strings. Each of these string-date objects is a node.

Stage 2: Construct Directed Acyclic Graph

For each node a

For each node b preceding a in a *(time period which you determine)*

1. Connect a to the b which is *(most similar)* to a
2. Assign edge weight based on *(similarity metric)*

Just to give you an idea of the adaptability of this design, I included the format *(idea)* to point out that a simple “out of the box” machine learning function could perform this function for you. To take this a step further, this system could be implemented with each of these ideas as a machine learning function whose hyperparameters would then be relevant to the programmer.

Stage 3: “Given the Directed Acyclic Graph with edge weights, delete a set of edges of minimum total weight so that each of the resulting components is single-rooted” (Leskovec et al). Each component graph resulting from the partition is a “narrative.”

**Economics Analysis:**

This system lends itself well to future development because of its modularity and parallelizability. While other graph and nonparametric methods can be the basis for another system, this approach guarantees that the investor will reap the full benefits of future improvements in parallel computing. This division of stages into logical steps is just a choice intended to simplify the overall idea of the system for a future developer. The system design has vast potential for improvement so some investors might find it worthwhile.

Investment in mental effort by computer scientists in this specific area (e.g. improving this framework) probably has high returns. This specific task has very good solutions which are short (speculation?) but require creativity. Creating a good NLP system to produce useful trading statistics is an ongoing tournament-type competition in the market with extremely high reward for being the best presumably.

**Future Work:**

In the future we hope to tune our system until it can generate trading stats with high predictive power. An “inversion” of this program into a risk-management tool might make it more useful (e.g. detect news before the median investor). But that is much less likely to succeed because of the speed advantage some professionals have. In the steps that were described, anything marked with a *(something)* is an easy start to improving the algorithm. These are all tasks that can be
improved by implementing open source functions like neural nets or random forests and tuning their parameters. A major current hurdle is access to data and the delivery of that data to the system (must be a real-time process for monetization of functionality).

Citations:
