Deriving Unsupervised Fluid Intelligence From Schema-Less Polymorphic Unstructured Data

Animesh Koratana
Department of Computer Science, Johns Hopkins University, Baltimore, MD 212180
National Security Agency, United States of America

Abstract

This study investigates the fluid intelligent capabilities of a layered artificial intelligence in the feed of streaming polymorphic unstructured data. With a data independent design, we are able to find relations between ideas within certain defined topics and the relationship between those ideas. Our algorithm was back tested against the United States Financial Markets as a topic with apparent and easily quantifiable relationships. We present a method to simulate multi-layered memory as a means to augment temporal relationships between nodes, define properties of node types without a predefined schema, and find centrality and distribution overlaps between the context sensitive surroundings of each idea node. We then present an in depth analysis into the errors with such a fluid intelligent machines and the simulacra that facilitate its fluid intelligent capabilities.

In a pursuit towards a truly dynamic artificial intelligence, it is necessary to establish a recurrent method to decipher the presence of concrete yet abstract entities (ideas) independent of a related and coherent topic set. A considerable amount of work venturing into this field has culminated in the prevalence of statistical methods to extract probabilistic models dependent on large amounts of unstructured data. These Bayesian data analytic techniques often result in an understanding superficial in the context of a true relational understanding. Furthermore, this bag-of-words approach when looking at amounts of unstructured data (quantifiable by correct relationships derived between the idea nodes) often relate to a single dimensional understanding of the topics at hand. Traditionally, when these topics are transformed, it is difficult to extract hierarchy and queryable relations using matrix transformations from a derived data set (McClelland *et al.* 1995). This project is an effort to change the approach from which dynamic fluid intelligence is derived, finding a backbone in streaming big data. Ideally, this model would be able to take a layered, multi-dimensional approach to autonomous identification of properties of dynamically changing ideas from portions of said data set. It would also be able to find types of relationships, ultimately deriving a set of previously undefined relational schemas through unsupervised machine learning techniques that would ultimately allow for a queryable graph with properties and nodes initially undefined.

Fluid Intelligence

Fluid intelligence can be defined as the capacity to think logically and solve problems in novel situations, independent of acquired knowledge. Psychology has found the basis of fluid intelligence in the juxtaposition of layered memory and application as means to essentially connect two fluid ideas with an an abstractly analogous property (McClelland *et al.* 1995). Such a mathematical design would have to be able to therefore derive temporal relationships with weighted bonds between two coherently disparate concepts through the means of similar properties (Jaeggi *et al.* 2008). These properties within node types will have to be self-defined and self-propagated within idea types (Kane *et al.* 2005).

A Node

Mathematically, a node can be defined as a statistically significant container of properties which has measurable relations with other nodes. In the context of a graph, an idea would be a node into which and from which relationships would be made and broken. Statistical significance will be approached in a more deterministic and dimensional method through which temporal memoranda and layered memory can be simulated. Ultimately this layering will allow for a more dynamic approach in the compartmentalization of idea types, and the generation of ideas. This design would require three components of implementation:

1. Finding statistically significant node types and nodes.
2. Finding property types for each of the nodes and populating the properties.
3. Finding the relationship types and the temporal relationships between nodes.

Node Generation

This will consist of all of the steps necessary to find statistically significant node types. It will first derive statistically significant node types and then derive (not project) a schema on the clusters of nodes. We will essentially be deriving schema from a schema-less design.

Idea Disparity

The process of node generation from unstructured data requires a foundation to find statistical distributions of words of a set A consisting of each of the documents aggregated. The dynamic set A will be a finite and elastic set of documents that will serve the purpose of representing the first layer of temporal memory without any sub-categorizations. Using a hybrid version of the collapsed Gibbs sampler, we are able to integrate out a set of variables into which we can assign distributions of words. Hierarchical Bayesian models yield multi modal distributions of words (Gilks and Wild 1992; George and McCulloch 1993).

$$g\left(x\right)p_{xy} = \frac{1}{d}\frac{g\left(x\right)g\left(y\right)}{\sum A \in \theta : A_{xy} \sim g'\left(xy\right)} = g\left(y\right)p_{yx} \quad (1)$$

This bag-of-words approach allows us to view the words of each subset distribution as statistical members of a larger set rather than lexical members of a semantic set. The equivalence is set up as *x y* between a permutation of possible node types. We begin with tokenizing the documents within A as inputs to our Bayesian Gibbs sampler. As an initial dimension to work off of, the derived distributions function similarly to those generated by the Latent Dirichlet allocation methods (LDA) (Gray *et al.* 2003). We use the LDA model used by Koratana (Koratana *et al.*) to find topic distributions in social media data. In essence, this approach is a hybrid of the LDA classifier method. Instead of topic distributions, we are able to find probabilities of each word given each node type. The sampler is able to find the following conditional probabilities using the bag-of-words approach in which each word is viewed as a statistical element within a vocabulary rather than a coherent part of a larger context (Porteous *et al.* 2008).

$$P(A_{(x,y)}|A_{-(x,y)}, (Z \in A); \alpha, \beta) = \quad (2)$$

$$\frac{P(A_{(x,y)}, A_{-(x,y)}, (Z \in A); \alpha, \beta)}{P(A_{-(x,y)}, (Z \in A); \alpha, \beta)} \quad (3)$$

In the figure above, we demonstrate the hybrid Latent Dirichlet allocation classifier as it find the probability of a statistical element within a subset, Z, of the populations set of documents, A.

Each significant subset, Z, of our document collection, A, now becomes a contender for becoming a node within our graph.

Unsupervised Multi-Network Training

The topic distributions of the current snapshot of nodes (of intermixed types) are then forwarded to an unsupervised neural network with a range of 10-20 hidden layers. A flexible preconditioned version of the conjugate gradient backpropagation method (Gray *et al.* 2003) is used:

$$\alpha_k = \frac{p_k^t}{p_k^T A p_k} = \frac{p_k^t(r_{k-1} + Ax_{k-1})}{p_k^T A p_k} = \frac{p_k^t r_{k-1}}{p_k^T A p_k} \quad (4)$$

α is the next optimal location vector relative to its position in the gradient of the linearized distribution sets, where the trained value would be a set of vectors of magnitude determining the distance of each distribution from the others from the subset. The hybrid gradient descent algorithm helps minimize the cross-entropy values during its classification. A separate and adequate network is trained and maintained for each subset of the original document set. The distributions with the greatest distance are then passed to another unique clustering algorithm based around minimizing the DaviesBouldin index (Hastie *et al.* 2009) between cluster components but still maintaining the statistical significance between cluster distributions derived in the LDA phase.

$$DB = \frac{1}{n}\sum_{i=1}^{n}\max_{j\neq i}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right) \quad (5)$$

Where n is the number of clusters, c_x is the centroid of cluster x, σ_x is the average distance of all elements in cluster x to centroid c_x, and is the distance between centroids c_i and c_j.

Schema-Less Design

At this point we are in the process of deriving schema from a schema-less design. However, when continuing to label the components of each of our distribution, we must ensure the testing of such a systems fluid intelligent design by means of designing a system independent from the content types and attributes of specific data set. A fluid intelligent design would ideally be able to find relationships between entities that are self-defined within the constructs of a predefined idea. Therefore, we are limited in the ability to predefine a structure of names and hierarchy into which to collapse the clusters derived above (He and Singh 2008).

Labeling At this point, we have a set of clusters of topic distributions derived from our original set A of documents (Engle *et al.* 1999). Each cluster has a node type (eg. stock, song) that is found using an association regression. Essentially, node type names are found looking for nouns that are commonly associated with a particular node type. This is common practice to perform the widening conversion from contents of a topic to the topic itself. After the clusters are derived, a parser labels the clusters of distributions with a particular node type. We now have a name associated with each of the topic clusters.

"Stock" $\subset (Z_1 \in A)$
"Industry" $\subset (Z_2 \in A)$
"Sector" $\subset (Z_3 \in A)$
"Exchange" $\subset (Z_4 \in A)$

Within each node type lies a set of distributions which now need to be identified for relationship aggregation.

Hybrid Named Entity Recognition Simultaneously, as we have derived names of topic clusters, we need to derive names of each of the distributions that fall under the labeled node type. As an extension of the finance metaphor, we have defined the existence of the "stock" idea and now need to identify that the ticker AAPL is a type of stock. Standard named entity parsers aim to classify elements into **predefined categories**. However, as the schema of the data is

Figure 1: Each circle here shows a derived latent distribution that falls under the node type of Industry (limited to 5 for visualization purposes). We now need to assign names to each of the distributions and identify properties for each of the distributions so an interpretable hierarchy can be created.

undefined in this construct, we are unable to train a named entity parser to identify the attributes of each distribution. In order to correctly identify un-trainable attributes of a derived distribution, we augment the named entity parser with the attributes of the unsupervised network trained to classify nodes. Embedded within this network which essentially founded the characteristics of the subset distribution, are common words associated with each of the clusters along with its distribution. We are able to compare the top 10% of distributions to feed to train a unique named entity parser (Etzioni *et al.* 2005). A formal evaluation of the parser produced near human performance with an F_1 score of 87.3%. An F_1 score is the harmonic mean of of precision and recall.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{6}$$

It is necessary to calculate sentence dependency trees from the tokenized words before feeding it into the named entity parser. For this, we use the Recursive Neural Tensor Network (RNTN) (Manning *et al.* 2014) semantic model developed by Klien and Manning at Stanford University. A sentence's logical form will consist of an interdependent tree of tokenized words. This model will compute compositional vector representations for phrases of variable length and syntactic type. The vectors error to propagate the RNTN can be defined as:

$$E(\theta) = \sum_i \sum_j t_j^i \log y_j^i + \zeta \|\theta\|^2 \tag{7}$$

This equation is used to provide a feedback to the RNTN about how good the vector representation of the dependency tree actually is. This is used by the Stanford NLP RNTN to propagate itself. The derivative for the weights of the softmax classifier are standard and simply sum up from each nodes error. We define xi to be the vector. at node i. We skip the standard derivative for W_s. Each node *back-propagates* its error through to the recursively used weights V, W. This method of backpropogation is independent of error alignment of the other NER and other trained networks. Let $\delta_(i, s) \in R_d \times 1$ be the softmax error vector at node i (De Marneffe and Manning 2008):

$$\delta^{i,s} = (W_s^T (y^i - t^i)) \bigotimes f^{'}(x^i) \tag{8}$$

High Level Negation These vectors can be reorganized to form relational trees of tokenized words. With this relational tree, it will be possible to negate portions of a sentence in context relative to another portion of a sentence. For example, in a sentence consisting of two independent clauses separated by a *however*, a dominance should be attributed to the clause following the conjunction. Variations in natural language like these minutely vary the overall meaning of each sentence but may affect the semantic structure of a particular sentence much more. The keyword dependent system would be heavily skewed in the presence of sarcasm and other complex linguistic systems (De Marneffe and Manning 2008). A

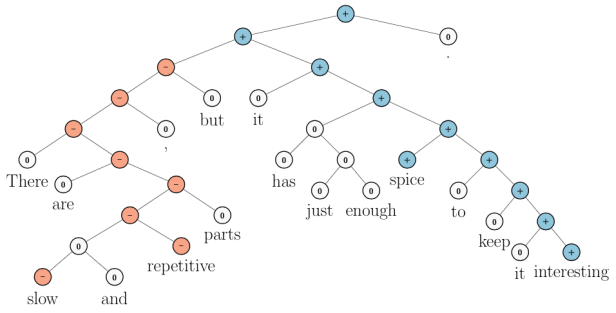

Figure 2: Here we have a parsed sentence based on the semantic structure of the sentence itself by the Stanford RNTN. We digress from the Bayesian statistical approach when trying to identify general names and identifiers for clusters along with property names and property values. (De Marneffe and Manning 2008)

relational tree constructed of the tokenized words would also allow for negation based on hierarchy of the word. Words with a higher relative location in the sentiment tree would be assigned higher weights in determining the overall sentiment of the sentence. Using properties of high level negation, we can derive properties of each of the node types by assigning a similar process to each cluster. For example, we can find the a property of the "stock" "AAPL" is "name" which will be assigned the value "Apple Inc.".
We now have a set of clusters. Each cluster is assigned a node type name, and each node within the cluster has an identifier and property key. Each node is queryable by its properties. A schema from the unstructured data has been autonomously created using unsupervised techniques.

[
"Stock": [GOOG, AAPL, ...]
"Industry": [Integrated Oil Companies, Publishing, ...]
"Sector": [Public Utilities, Health Care, ...]
"Exchange": [NASDAQ, AMEX, ...]
]

Each entity here is a self-populated derived value from our schema-less design. We have identified the existence of a stock type node, found that AAPL is a type of stock, and that a property of the stock ticker AAPL is a name which is Apple Inc.

Cognitive Inventory In order to simulate temporal memory, we do not commit each iteration of these derived distributions to database. Instead, we assign temporal weights to propagate the existence of each node, strengthening its bonds by the persistence of the node over time (Kacholia *et al.* 2005). If, for example, the ticker AAPL remains present for a longer period of time, the stock becomes committed to a database instead of temporarily finding its way onto the relational schema. Therefore, **as memory becomes reinforced as a function of time, we begin to see more accuracy in the persistence of correct nodes and identifiable ideas**.
We do not expect the debut of a new node to remain present unless the data that the engine is aggregating consistently finds the node as statistically significant for a continued amount of time.

At this point we have derived statistically significant node types and found property types associated with each of the node types. Each node has been populated with the appropriate values and we have established the clusters within each node type.

Figure 3: The industry nodes found above in *Figure 1* now have been assigned names and a schema has been derived for the node type. The same process has been followed for each of the other subset distributions or node types.

Relationship Extraction

The process of relationship extraction depends on the preemptive steps laid out by the node aggregation steps. We count on the fact that a schema (although dynamic)[1] has been laid out and use the properties of the distribution clusters to layer relationships.

Distribution Overlap

We assume that **initially** inter-node type relationships cannot exist. That is, a stock cannot directly be related to another stock. We must begin by finding relationships between node types (find that AAPL *stock* with all of its properties relates to the Computer Manufacturing *industry*).
In order to do this, we begin by classifying the streaming documents and associating them to nodes. This entails using another Named Entity Parser [2] that is trained to identify mentions of our derived nodes. At this point, we have already found the predefined categories (our nodes) and therefore we are able to easily associate documents (articles, tweets, posts, and blog entries) to each associated node type (Etzioni *et al.* 2005). Completely separate (but similar in process) from the initial hybrid Gibbs sampling method, we assume large amounts of data when we derive distributions of topics surrounding each node. A hybrid Latent Dirichlet allocation (LDA) method is used as we derive α (the K-dimension vector of positive reals) from the initial distribution of the cluster. We can assume this α value is approximately accurate because embedded in the sampler for each of the cluster distributions are the statistical characteristics of the new node cluster. It is necessary to mention, we have at this point completed the first dimension of linearized vector spaces (each representing a node) are are now working on the second to K^{th} dimensional spaces of the temporal memory clusters. Therefore, the cluster of documents we reference for each of the individual nodes in this case is different from the larger cluster of similar node types. Instead, we are now working with individual nodes and clusters of documents unique to each node (looking at the documents that reference AAPL).

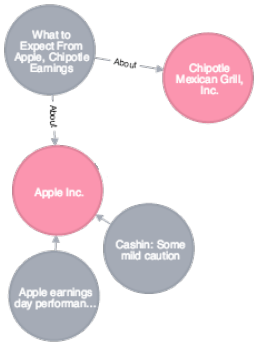

Figure 4: This shows how the documents (articles in this case) are associated with each of the nodes. The same process was followed for each node that was derived.

We run the collapsed Gibbs sampler with the intent to model each distributional characteristic of each node cluster. We use the previously established method to find the probability of a word j within a subset Z of a vectored representation θ of the original document corpus A surrounding the node cluster within a K dimensional vector field.

$$\int_{\theta} P(\theta_j|\alpha) \prod_{t=1}^{N} P(Z_{j,t}|\theta_j) d\theta_j \tag{9}$$

$$= \int_{\theta_j} \frac{\Gamma(\sum_{i=1}^{K} \alpha_i}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} \theta_{j,i}^{\alpha_i - 1} \prod_{t=1}^{N} P(Z_{j,t}|\theta_j) d\theta_j \tag{10}$$

At the end of this step, we have topic distributions for each of the node clusters and we need to begin comparing each of the distributions between nodes.

[1] Having a dynamic schema allows us to maintain fluid intelligent capabilities (as the potential for new ideas is always open as the data changes). Since the relationships that will be derived here depend on the schema, a changing schema proves to change the substance behind any of the relationships we will derive.

[2] This NER parser is **separate** from the one trained to identify cluster distribution labels.

Distribution Similarity We train an unsupervised neural network to identify similarities between topic distributions. However, this time instead of comparing probability vectors, we compare the derived topic distributions themselves.
As an input function to serve as a tool for the network, we define the localized calculation of lexical (equivalently the Levenshtein) distance between any two words. This essentially is the number of lexical transformations that are needed to change string a to string b assuming $a \neq b$. The lexical distance is compared against a threshold that is used to calculate distance between distributions of disparity node types.

$$lev_{a,b}(|a|,|b|) = \begin{cases} \max(|a|,|b|) \\ \min \begin{cases} lev_{a,b}(|a|-1,|b|)+1 \\ lev_{a,b}(|a|,|b|-1)+1 \\ lev_{a,b}(|a|-1,|b|-1)+1_{(a\neq b)} \end{cases} \end{cases} \tag{11}$$

After the topic distribution are derived for each of the nodes, we are able to compare the distributions using the trained unsupervised neural network. We pass topic distributions and allow the network to use lexical distance as a network parameter within its hidden nodes to calculate a similarity index.
For example, to make the connection between the AAPL stock and the Computer Manufacturing industry, we compare the distributions derived around each of the nodes.

Table 1: Here we can see that topic 2 of Computer Manufacturing is similar to Topic 1 of AAPL node. We can deduce a relationship between the two nodes.

AAPL			**Computer Manufacturing**		
Topic 1	Topic 2	...	Topic 1	Topic 2	...
computer	carl icahn	...	cpu	windows	...
macintosh	investors	...	ibm	macintosh	...
...	...	...		...	...

Eventual Consistency This relationship can be committed to a temporary memory in a temporal dimension as its relationships are primed as a function of time. Therefore, as the similarity between AAPL and Computer Manufacturing is continually being found for an extended period of time, the relationship between the two nodes will grow stronger before it is committed to the graph database where the relationship will stand until it is broken. We run this similar process permutationally between distributions of nodes to **concurrently** find the relationships between all nodes. The distributions are recalculated in the presence of newly attributed data since Gibbs sampling is computationally intensive and futile to run in real time. After 48 hours of running this algorithm, we were able to correctly identify the relationships between our derived nodes.

Centrality

Once a schema and relationships are derived, it is easy to find measures of centrality using adjacency matrices to find

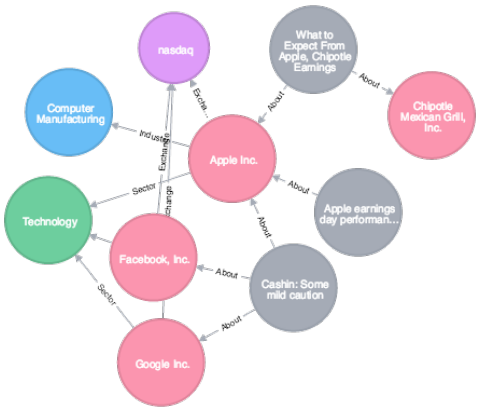

Figure 5: A derived graph surrounding the node of AAPL. Notice the relations between the node types.

eigenvector centrality. For a given graph $G := (V, E)$ with $|V|$ number of vertices, let $A = (a_{v,t})$ be the adjacency matrix, (for example $a_{v,t} = 1$ if vertex v is linked to vertex t), and $a_{v,t} = 0$ otherwise. The centrality score of vertex v can be defined as:

$$x_v = \frac{1}{\lambda} \sum_{t \in M(v)} x_t = \frac{1}{\lambda} \sum_{t \in G} a_{v,t} x_t \tag{12}$$

This concludes the theoretical portion of our exercise.

Analysis

As this engine functions as a completely unsupervised system, we cannot follow the traditional method of error tracing. We need to calculate algorithm errors for a number of steps within the process.

1. Error of the Hybrid Gibbs Sampler used to set up the dynamic equivalence relations
2. Error of the LDA classifier used to find node distributions
3. **Average** error of the unsupervised multi-networks used for classification and categorization of nodes into clusters
4. Error of the hybrid named entity parsers used for labeling nodes and property types
5. Error of the unsupervised network used to find distribution overlap of the node sub-distributions

Hybrid Gibbs Sampler (+LDA) Error and NER Error

It is trained to predict series $y(t)$ given d past values of $y(t)$ and another series $x(t)$. The sampler is forced to find the distribution of vectors spaces between dimensions in order to simplify our problem of categorization on a schemaless base. Again, as our hybrid Gibbs sampler is trained, we are able to see that the recursion of error remains relatively constant (usually unlike a sampler) (Blei *et al.* 2003; 2001). This is because we are able to train our sampler on

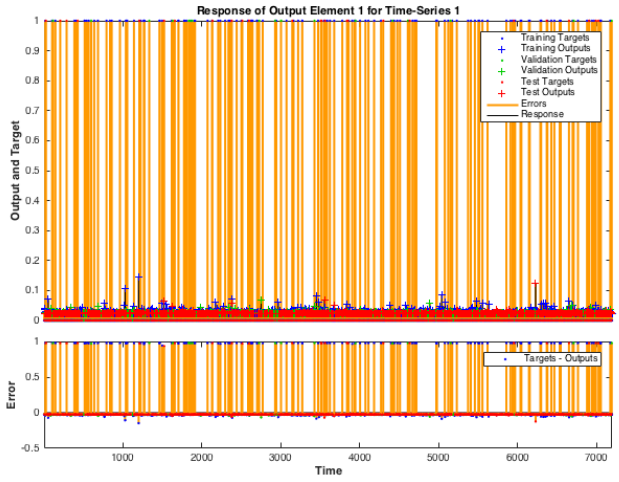

Figure 6: We find that the density of error within the Gibbs sampler is relatively constant.

dynamic changing data sets (aforementioned set A is dynamic as other subsets $Z \in A$ are static). The error propagation of the hybrid sampler is keen to changes within the distribution as the unsupervised network is tuned to listen to changes, rather than the content of each derived distribution. In order to classify the error of the hybrid Gibbs sampler, we classify our algorithm's problem definition as a nonlinear autoagressive with an exogenous input (NARX) system.

Unsupervised Multi-Network Errors

The gradient based back propagation allows us to compare (arbitrarily) the correlation between the input and error between matched distributions. It is important to notice that these do not represent the distributions themselves, but rather the correlations between the input semantic distributions and the vector error within the K dimensional vector spaces normalized within the confidence limit of our sampler.

An inverse relation between error and the significance of the distribution at hand is important to witness as it propagates correct clustering in the context of our schema-less distributions.

We now continue to observe the error histogram of the clustering mechanism of the found unsupervised network. Here we find an **average** of the scaled conjugate gradient back propagation method errors. Most of the errors are centered around 0. This error distribution accounts for the errors found in both types of the unsupervised network training algorithms.

In order to gain better insight into the workings of the clustering mechanism of the unsupervised neural network, we need to traverse the epochs of the topology of the clustering mechanism.

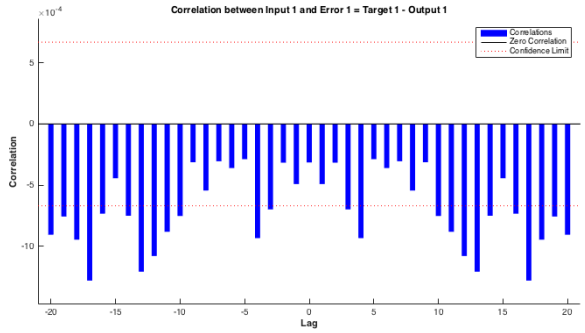

Figure 7: The correlation of error propagation change inversely in tandem with the distribution significance. This is a favorable trait we look to propagate, because as the distribution's distances increase (it lexical error), we want the ability of the unsupervised network to determine there distances to increase in tandem.

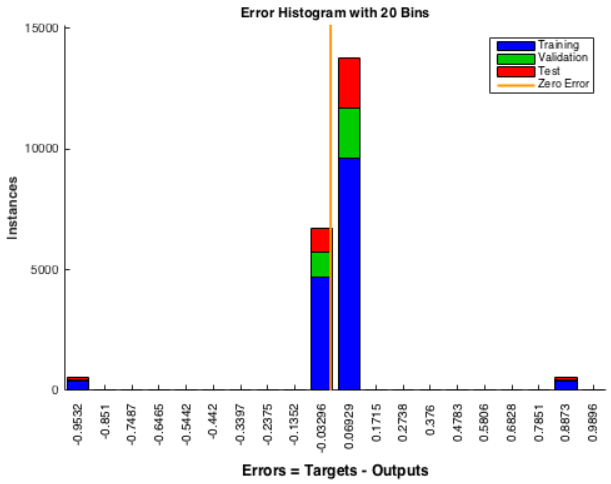

Figure 8: Here we witness the general error of our multi-network surrounding the 0 error bar in orange.

Compositional Error

It is necessary to identify the strengths and weaknesses of such an engine. We observe the capability of using statistical methodologies in combination with hybrid unsupervised machine learning algorithms to derive **lexicographic** associations between derived topics and ideas represented by nodes. However, it is necessary to realize the dependence of such a system of lexicographic languages. Since this is dependent on both statistical and semantic parsing algorithms, we, at this point, cannot safely assume compatibility with non-lexicographic languages. The pre-trained systems used in this engine is only the Stanford Recursive Neural Tensor Network (RNTN) network. Every other system is dependent on the initial aggregation of correct distributions. As the algorithm find its foundation in statistical methods, it is as-

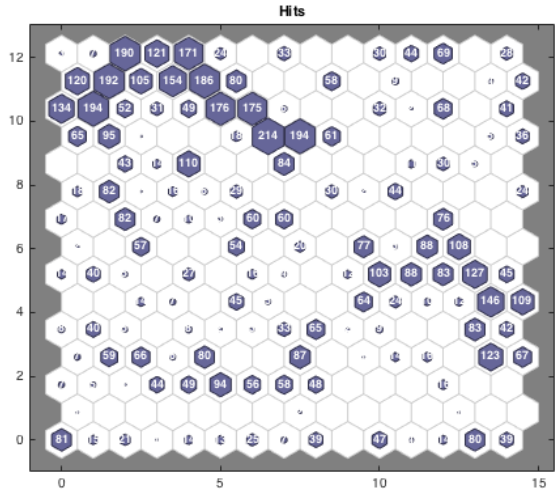

Figure 9: The topology of the layers of the clustering networks. We can see similarity indexes being assigned temporal primers to solidify relationships and associations made in this phase.

sumed to be largely accurate. However, error projects on a interdependent systems making the effect of any error at an early stage magnify at stages of NER recognition and association matrices.

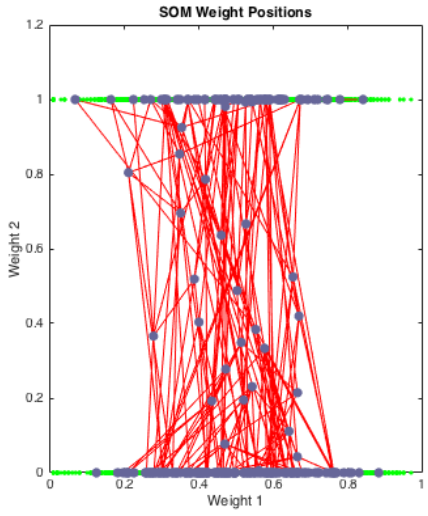

Figure 10: This is a visualization of the weight transformations from the softmax (SOM) error of the multi-network to the named entity parser used in the subsequent steps.

Discussion and Conclusion

In this exercise, we were able to create a method to extract fluid intelligence from a polymorphic data source. From a dynamic input of data sources, we were able to derive a schema from a schema-less unstructured source using statistical methods in unison with unsupervised machine learning algorithms. This is an approach not pioneered in the history of eventually consistent memory: we were able to derive temporal relations between derived, schema-less nodes, similar to how a human can derive explicit relationships from implicit and unstructured data. The layered and temporal relationships, tempered with a persistence over time, served to reinforce the memory in a multidimensional vector space, ultimately making its way down to the initial two dimensional relationship we perceive as quantifiable. Centrality between nodes could be derived on a node specific level as the perspective of the temporal relationships changed based on the nodes. This opened the door to perspective based centrality where each node was central to individual ideas, and changes in values of centrality could effect the necessity of such nodes on the general relationships found in the path between two nodes.

The algorithm's inherent dependence on lexicographic languages is a key limiting agent to this engine. We cannot bank on parametric or statistically recursive characteristics of language (commonly found in English and other Germanic languages) as a means to quantify schema.

The fluid intelligent capabilities are highlighted in the fact that the nodes and relationships derived by the engine are all **queryable** and can be used to interface with any third-party software which can interface with a graph database. In total we aggregated **9000 nodes** in a time of **72 hours** with a total

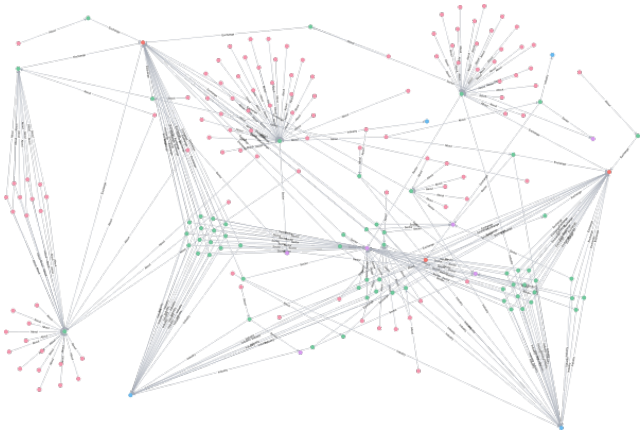

Figure 11: As a thought experiment, we display 100 nodes (less than 1% of the total number of nodes) found and derived by this engine. We can look at these as relational (common), or from the perspective of sub-latent nodes. The next step would be finding relationships between derived engines and applying fluid intelligent capabilities to extend the assumption and reactive capabilities of a future artificial intelligence.

of **1.7 million relationships gathered** (although constantly dynamic).

The dynamic intelligence such an engine can offer in the presence of scalable and ubiquitous dynamic unstructured data is countless. Such and engine has ranges of application from medicinal propagation and relation, to curbing cyber-terrorism by relating attacks to influxes in data in dark domains. It is necessary to realize that this engine was not designed with any particular dataset in mind (just a large amount of unstructured data) and therefore can be applied to any such data source. The metaphor of financial markets was easily identifiable as a topic that has easily quantifiable relationships and easy to test for the purposes of testing such an engine. The schema and data independent design of this fluid intelligent engine is what separates this from other static intelligent extraction techniques. The engine can very well be applied to more inquisitive locations to find relationships between ideas unknown to humans themselves.

Acknowledgments

The author would like to thank Dr. Celia Rhodes of Stanford University for her technical guidance and field expertise provided throughout all portions of experimentation and analysis.

References

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. In *Advances in neural information processing systems*, pages 601–608, 2001.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

Marie-Catherine De Marneffe and Christopher D Manning. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics, 2008.

Randall W Engle, Stephen W Tuholski, James E Laughlin, and Andrew RA Conway. Working memory, short-term memory, and general fluid intelligence: a latent-variable approach. *Journal of experimental psychology: General*, 128(3):309, 1999.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134, 2005.

Edward I George and Robert E McCulloch. Variable selection via gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.

Walter R Gilks and Pascal Wild. Adaptive rejection sampling for gibbs sampling. *Applied Statistics*, pages 337–348, 1992.

Jeremy R Gray, Christopher F Chabris, and Todd S Braver. Neural mechanisms of general fluid intelligence. *Nature neuroscience*, 6(3):316–322, 2003.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *Unsupervised learning*. Springer, 2009.

Huahai He and Ambuj K Singh. Graphs-at-a-time: query language and access methods for graph databases. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 405–418. ACM, 2008.

Susanne M Jaeggi, Martin Buschkuehl, John Jonides, and Walter J Perrig. Improving fluid intelligence with training on working memory. *Proceedings of the National Academy of Sciences*, 105(19):6829–6833, 2008.

Varun Kacholia, Shashank Pandit, Soumen Chakrabarti, S Sudarshan, Rushi Desai, and Hrishikesh Karambelkar. Bidirectional expansion for keyword search on graph databases. In *Proceedings of the 31st international conference on Very large data bases*, pages 505–516. VLDB Endowment, 2005.

Michael J Kane, David Z Hambrick, and Andrew RA Conway. Working memory capacity and fluid intelligence are strongly related constructs: comment on ackerman, beier, and boyle (2005). 2005.

Animesh Koratana, Mark Dredze, Margaret S Chisolm, Matthew W Johnson, and Michael J Paul. Studying anonymous health issues and substance use on college campuses with yik yak.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577. ACM, 2008.