An Articulation of Natural Language for Forecasting: Next Steps in the Evolution of Relevance

Kenneth Luke Skertich

BSc Computer Science

2708233

School of Computer Science

College of Engineering and Physical Sciences

University of Birmingham

2024-25

Abstract

Forecasting models often become unreliable during volatile periods caused by anomalous events, from COVID-19, earthquakes, or even sports championships. Recent efforts have shown that supporting these models with external information about the world can significantly boost performance. However, there is no consensus in literature about what information in the world is relevant for the forecasting task. Without clear definitions, the current literature fails to address the core natural language representation trade-off between noise and language nuance.

We formalize the role of natural language in forecasting by introducing a new framework for defining relevance and utility. Then, to operationalize new definitions, we introduce Task-Aware-Covariates (TAC) generated from a Generative Adversarial Network (GAN), a novel utility-optimizable natural language embedding framework for forecasting. We define the most relevant information as that which improves the downstream task.

TAC-GAN uses a transformer-based GAN to encode Wikipedia page embeddings, guided by two objectives: learning the relationships between the events of a day through a reconstruction task, and minimizing forecasting error on the quantity prediction.

We introduce an original mathematical underpinning of time series volatility to bridge theoretical gaps in multi-data stream forecasting. Our approach reframes forecasting under volatility as a learnable optimization of relevance itself, offering a principled foundation for a new class of models.

We apply TAC-GAN's learned embeddings as covariates to an LSTM model forecasting the S&P 500 index for proof-of-concept, demonstrating improved performance over GAN-Event LSTM under anomalous conditions.

Keywords: TAC-GAN, utility, forecasting, natural language representation, embeddings

Acknowledgements

I would like to thank my supervisor, Dr. Rami Bahsoon, for his invaluable support and confidence in my ability to meet the challenge of this thesis.

Abbreviations

TAC	Task-Aware Covariates
GAN	Generative Adversarial Network
MSE	Mean Squared Error
wMAPE	weighted Mean Average Percent Error
LSTM	Long-Short-Term Memory
RNN	Recurrent Neural Network
REN	Recurrent Event Network
NTN	Neural Tensor Network
TF-IDF	Term Frequency-Inverse Document Frequency
BoW	Bag-of-Words
NLP	Natural Language Processing
IE	Information Extraction

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

In many domains, people are far more surprised when forecasts are wrong than when they are extremely precise. Forecasts inform us when to make a decision, such as cancelling plans when meteorologists warn of a storm. In some contexts, these decisions shaped by forecasts can have major consequences, where even a small error can be costly in domains like supply chains (e.g. warehousing) or public health (e.g. pollution).

Forecasting models traditionally rely on stable market patterns and historical data. However, anomalous events–such as global pandemics, political shifts, and sudden consumer trends–can disrupt these models, leading to inaccurate predictions and costly supply chain inefficiencies. Data that exists in multi-causal systems is particularly at risk during anomalies.

Predicting quantities during unprecedented circumstances still proves difficult, where even state-of-the-art models fail to maintain accuracy. To handle volatility, newer frameworks incorporate exogenous data, called covariates. If the goal is to predict rainfall, raw rainfall data could be supplemented with the melting rate of upstream glaciers. With data in multi-causal systems (e.g. as government bond purchase rates, produce prices, etc.), selecting the exogenous data becomes far more involved. Research often turns to natural language to model the complexity of the world. This brings us to Natural Language Processing (NLP).

Given the high-dimensional nature of natural language, NLP typically maps language into a geometric manifold called embedding space. We'll call the extraction of information from embedding space 'representation'.

Representation concerns selecting and structuring information to optimize its relevance for deep learning forecasts. The requirements of relevance change with domain shifts and even timeframes; there is no universal definition of relevance. Without definitions in place, current literature implicitly balances between maintaining semantic nuance (embedding space complexity) and reducing noise (information that can hurt the forecast accuracy). We argue this trade-off is a central challenge in forecasting with natural language-based covariates.

To overcome this challenge, we propose that relevance is not an intrinsic property of the exogenous data. Relevance should be evaluated on the basis of how information improves forecast performance. This defines our notion of relevance as task utility and forms the basis of Task Aware Covariates (TAC), which are learned through a Generative Adversarial Network (GAN). The GAN learns to represent structural relations between world events (Wikipedia event pages) that occur each day, repackaging them as 100-dimensional covariates. Concurrently, the GAN also learns what kind of information improves the deep learning forecasters performance through a live feedback loop. Our deep learning model is a Long-Short-Term Memory Network, a type of Recurrent Neural Network (RNN).

The goal of this report is not to beat all existing models. Rather, this is a scientific investigation of the nature of forecasting itself, which evolves into our TAC-GAN-Event experimentation.

1.1 Problem Statement

The basis of the problem:

In forecasting literature, deep learning models are supplemented with external signals to contextualize time series data that shows no discernable patterns. Natural language can come from a variety of sources and raises potential issues with verisimilitude and noise. Research shows that noise can be reduced with careful dimensionality of data, but this can come at the cost of loss of complexity.

The problem:

Stricter semantic filtering loses higher-order relations, making forecasts erratic; looser filtering admits noise, degrading performance.

1.2 Contributions

Theoretical contributions:

- Factor Influence Distribution, a new definition for multi-causal systems to explain volatility.
- A new definition of information relevance as measurable utility toward a downstream forecasting task, addressing the lack of consensus in prior literature.
- We reframe the commonly held opposition between semantic complexity and relevance as a learnable optimization problem.
- We show empirical evidence that Wikipedia event page content and page views are only moderately correlated over time. We also discuss the demographic bias introduced by page views.
- We show how defining a global point of relevance in the embedding hypersphere is undermined by a direction-meaning contradiction, which can undermine utility.

Applied contribution:

- We design a novel GAN architecture that includes a downstream feedback loop for domain-specific adaptation, called TAC-GAN-Event. The model outputs utility-aware embeddings between Wikipedia2vec embeddings of Wikipedia article pages.
- We introduce a GAN stabilization method (cosine angular noise perturbation) tailored to hyperspherical semantic embeddings, improving generator-discriminator dynamics during training.

1.3 Paper Structure

he rest of this paper is structured as a continuous probing of possible definitions and solutions under those definitions. Section 2 uncovers the explicit and implicit definitions in current literature while critiquing the assumptions that underlie them. Section 3 focuses on evolving existing definitions under a new set of assumptions, as well as the methodologies and assumptions that entail those definitions. Section 4 presents our IE process. Section 5 illustrates the greater adversarial architecture for the GAN. Section 6 provides the integration of a downstream-feedback loop into the GAN. Section 7 shows our experimental setup and results.

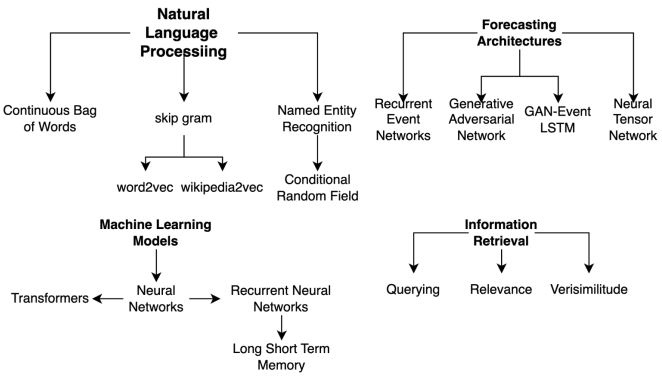

Figure 1.1: A hierarchical diagram of concepts mentioned in this report.

1.4 Key Definitions

Definition 1.1 (Representation). A representation RRR is a mapping from raw text TTT to an embedding space $\mathbb{R}^d$, chosen to balance (i) the amount of extraneous variance ('noise') and (ii) the preservation of task-relevant signal ('relevance').

Definition 1.2 (Information Extraction). For forecasting, information extraction is the natural language processing, typically involving three phases:

- **Selection**: choose which text units (words, sentences, articles) to include.
- **Transformation**: map them into a numeric form (embeddings, tuples, etc.).

- **Filtering/Weighting**: discard or re-weight features based on relevance criteria.

Definition 1.3 (Relevance). The IE criterion—semantic, statistical, or task-based—that guides the inclusion or weighting of features in RRR. When the guiding principle is *task utility*—i.e., the expected reduction in forecasting error—we call this *utility relevance*.

Definition 1.4 (Centrality Relevance). Point-based relevance measured by proximity $d(e, c)$ of each event embedding e to a fixed "central" point e in the latent space.

Definition 1.5 (Relational Relevance). Structure-based relevance measured by the **coherence** or **association** among all events in a set, as captured by a (learned) function $f(\{e_1, \ldots, e_n\})$.

Definition 1.6 (Factor Influence Distribution). The probabilistic influence of factors on a model's output. Let $\{F_1, F_2, \ldots, F_n\}$ be the set of n latent or external factors that drive a data generation process in a closed system. Each factor F_i can take values in some space Ω_i, and taken together they follow a joint probability distribution:

$$(F_1, F_2, \ldots, F_n) \sim P_F$$

Let X be the observed output (e.g., a time series) in $\mathbb{R}^m$ determined by a function g:

$$X = g(F_1, F_2, \ldots, F_n),$$

where $g : \Omega_1 \times \cdots \times \Omega_n \to \mathbb{R}^m$ delineates how factors combine to produce X.

For each factor F_i, define an influence function $\phi_i : \Omega_1 \times \cdots \times \Omega_n \to \mathbb{R}$ that measures how changes in F_i affect X. We then define the Factor Influence Distribution (FID) as the distribution of these influence functions under P_F:

$$(\phi_1, \phi_2, \ldots, \phi_n) \sim P_\phi,$$

where P_ϕ is induced by drawing $(F_1, F_2, \ldots, F_n) \sim P_F$ and evaluating each ϕ_i.

Finally, we get the partial derivative of the output with respect to factor F_i:

$$\phi_i(F_1, F_2, \ldots, F_n) = \left\| \frac{\sigma g(F_1, F_2, \ldots, F_n)}{\sigma F_i} \right\|$$

This FID characterizes how likely each factor is to significantly shift the observed data X.

Definition 1.7 (Utility Relevance). The notion that information is relevant if it improves the downstream task.

CHAPTER 2

Literature Review

This is a thematic literature review. It is broken into two main branches: models that (1) use only past quantities of the quantity we want to predict and (2) use the past quantities of the quantity we want to predict in addition to temporally aligned natural language signals. We refer to (1) as Single Data Stream Forecasting and (2) as Forecasting with Supplementary Data Streams.

2.1 Single Data Stream Forecasting

2.1.1 Traditional Forecasting

Statistical baselines (e.g. moving averages) have largely been surpassed by neural approaches [15], which capture non-linear, temporal patterns. Yet in many real-world systems, the most meaningful patterns emerge not from the data itself, but from its context. In highly interactive systems, where exogenous variables drive behavior, forecasting requires more than historical modeling. In this paper, we treat natural language as a proxy for those variables.

2.1.2 Generalizable Forecasting Models

One approach to tackling the shortcoming of deep learning for anomalous events is to amass a behemoth amount of data in a modular fashion, which Google has done. TimesFM by Google is a decoder-only foundation model trained on many different types of time series. A "decoder-only" Transformer omits the encoder component, relying solely on the decoder to handle the task. This means the model generates outputs based only on the input provided directly to the decoder, without an intermediate encoded representation. This architecture is particularly effective for autoregressive tasks, where each output is generated sequentially, conditioning on previous outputs [9]. Large language models like GPT-3 utilize this decoder-only approach.

Conceptually, TimesFM has learned the 'grammar of time series forecasting' [5]. Calling this deep learning model a 'grammar' is a comparison to automata theory, where a grammar is a set of rules that define the structure of a language. By training on many different kinds of datasets, from Wikipedia Page Views to traffic data, along with synthetic data, TimesFM learns the time-series patterns for almost any kind of data with 100 billion time points.

While it is an extremely powerful pre-trained model, it is nevertheless composed of only one modality of data, raw numeric values, with no external knowledge base. Single modality prediction is blind to causal volatility. The model is highly generalizable and can achieve state-of-the-art average accuracy across a broad set of time series datasets (Monash, Darts, Informer), but when faced with a unique market pattern during anomaly, the model is equally ill-equipped as an LSTM because, in isolation, the time series break their own patterns.

We turn to observing the system in which a time series lives. For example, stock market data can be influenced by global politics, company performances, technological development, etc. We call these factors, or variables. If occurrences relating to these topics were input to a forecasting model as factor inputs, their influence on the model is tractable. In machine learning, Shapley values and partial derivatives deliver point estimates or local/global averages of factor contributions.

Contrast this with our definition of Factor Influence Distribution that shows "influence" itself is a probabilistic property of the input space.

Intuitively, two datasets can exist in the same system and evolve differently according to both intrinsic properties (i.e. features) and extrinsic properties (relations to other data). Formally, consider that dataset A and B live in the same closed system with a set of factors (variables) $\{F_1, F_2, \ldots, F_n\}$.

Even in a closed system, the distribution of how those factors manifest as influential over time are different according to the relations set of all possible pairwise relations between the data. In machine learning, Shapley values and partial derivatives deliver point estimates or local/global averages of factor contributions. In contrast, our definition of FID makes "influence" itself a probabilistic property of the input space. As shown in Figure 2.1, these profiles collectively yield a different influence distribution for different datasets in the same system.

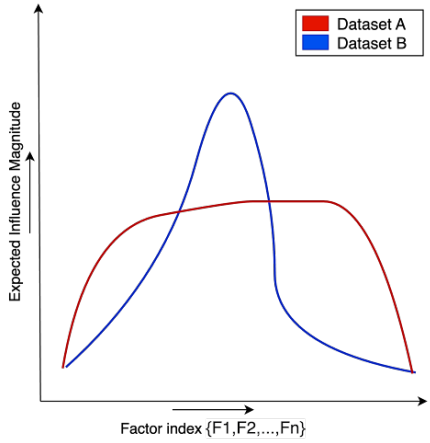

Figure 2.1: An arbitrary visualization of FID in a closed system, where dataset A and B live in the same closed system and are temporally dependent. In (a), we compress each factor's probabilistic influence into its average on the y-axis, $E\left[|\varphi_i(F_i)|\right]$.

Volatility alone doesn't fully explain why a dataset is difficult to predict. Definition 1.6 formalizes this insight with a distributional perspective that each factor F_i has a probabilistic influence profile. These profiles define how strongly and how often each factor affects the target dataset. In practice, systems are usually open, not closed, where it is impossible to obtain and process all the factor data effectively. Rather than relying on a handful of factors to explain all volatility, we posit that the data we do have can explain just a portion of the volatility for prediction over a target, and that portion is its utility.

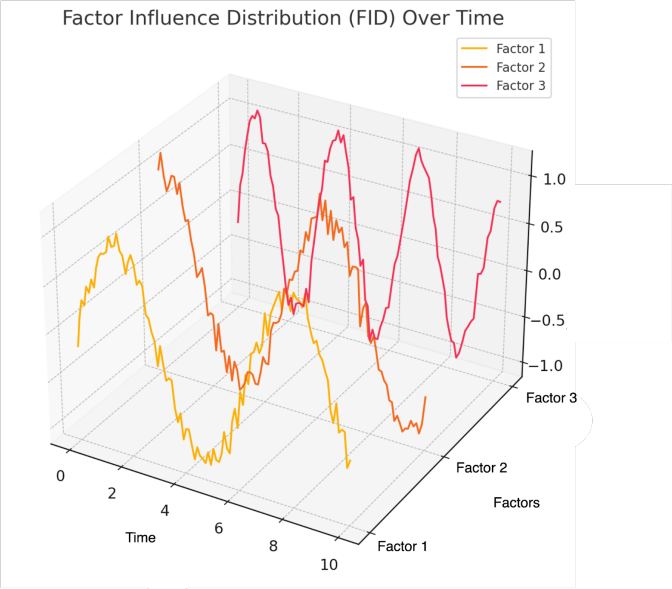

Figure 2.2: A 3D visualization of temporal FID.

While traditional interpretability methods provide local or global importance scores, FID treats influence as a stochastic quantity that varies with system state. This offers a more faithful view of influence in temporally dynamic or non-stationary systems.

We can only infer how many exogenous factors exist for the datasets, and their respective FID's. For now, we can use our intuition about these open-system datasets by asking questions like: how many factors are likely to contribute equally to the output? This theoretical framing invites a reinterpretation of existing benchmarks, such as TimesFM. Consider the following datasets evaluated by TimesFM.

Table 2.1: Example datasets used for evaluating TimesFM.

Dataset Source	**Dataset Name**	**TimesFM MSE**	**Fat-Tailed Probability**	**Rank Among Baselines**
Monash	tourism yearly	109977.29	High	14th of 16
Monash	traffic hourly	0.01	Low	1st of 16
Monash	covid deaths	209.80	Moderate	7th of 16
Darts	HeartRate Dataset	5.85	Moderate	4th of 9
Darts	Sunspots Dataset	41.40	Low	1st of 9
Darts	Wine Dataset	2871.33	High	4th of 9

In essence, when patterns in data are volatile, influenced by high-dimensional variables arising from world events or complex human behavior, we should not accept state-of-the-art results at face-value.

Much research has been done to integrate an external knowledge base as a secondary information stream to model these high-dimensional dependencies. In the next section we will explore recent literature on forecasting with a contextual data stream.

2.2 Forecasting with Supplementary Data Streams

We have established that traditional forecasting models–both deep learning and statistical–are inherently unequipped to handle the volatility of data impacted by world events. Now we will review existing literature, focusing on notions of how to supply traditional deep learning models with information about the world. Literature seeks answers to these primary questions:

- What information about the world is empirical?
- How can the processing of data alter verisimilitude?
- What information about the world is relevant?
- How should information be selected and structured to increase performance?

Through this literature review, we will uncover a central dilemma within event representation. The more information we preserve, the more noisy signals become. On the other hand, the more information is reduced in line with centrality criteria, semantic complexity is lost. The order of research does not imply an order of "worst to best" performance. Each successive paper discussed in this review conceptually leads to the next, each having their own gains, drawbacks, and domains.

Therefore, the applied and theoretical contributions of this paper, TAC-GAN-Event, is only a natural unification of otherwise conflicting notions of relevance and complexity; the idea arises from the evolution of the answers to the above research questions, landing on a notion of relevancy as "utility" for minimizing error on the forecasting task, while maintaining empirical data.

2.2.1 Labeled Event Representation (Relational Relevance)

This section focuses on literature that implements Definition 1.4 and the risks and limits of reducing natural language to labels for relevance.

Keyword Structure Representation

In 2024, Chakraborty et al. achieved promising results on produce price forecasting by utilizing an external knowledge base of millions of news articles from a major Indian newspaper [4]. Chakraborty et al. represented each day's events as a vector alongside quantities for prediction to create what they term a "Recurrent Event Network" (REN). Each day's vector is multidimensional, where each dimension contains a central keyword from an article (e.g. scam).

The chief novelty of this paper lies in the reduction of whole news articles to a few keywords representing meaningful and relevant events. The researchers assert that including meaningless events into a day-level representation can be detrimental to performance. One of the baselines is an LSTM + word2vec embedding. They claim that the poor performance of supplying the LSTM with entire news articles–with word2vec embedding model–hinders performance with noisy semantic event representations. Therefore, they propose the following methodology:

1. Restrict news article representation set to the vocabulary of all 'event trigger' words, as classified by Conditional Random Field. Trigger words are identified from the headline and first paragraph of the article.

2. Form clusters (using the elbow method for optimal number of clusters) within the semantic universe of event triggers, with the distance measure defined as the cosine similarity (Manning et al. 2008) between the word2vec representations of each element in the vocabulary.

3. Feed the labeled event data alongside daily quantities to train a Recurrent Neural Network.

4. As a bonus, the researchers then credit individual events with a causal relation to the prediction. In direct association with a particular data point, an event receives a "relevance score," derived from the gradient-based explanation method to explain a particular input's impact on the output of a Neural Network [4]. However, because the feed-forward process of a Neural Network is non-linear, these explanations provide a coarse-grained, probable set of candidate events that dictate price points in a given time frame.

This system provides impressive interpretability. Their custom embedding model also achieved 93% accuracy at identifying the central event of articles.

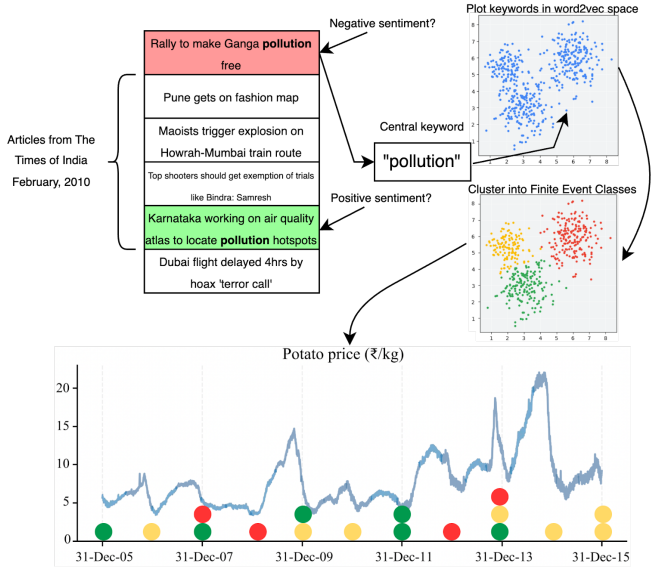

Figure 2.3: We provide an illustrative diagram of the NLP pipeline for REN embeddings: text → keyword extraction → plotting in word2vec space → clustering in word2vec space → supplying RNN with a daily vector of labels.

These researchers show that this model outperforms baselines for prediction over the 7 time series (e.g. onion price, potato price, etc.). We argue that the paper still has several shortcomings that hinder its potential for even better performance, specifically:

- Chakraborty et al. recognize that their news data comes from a "single albeit reputed source, which might raise questions about the accuracy or potential systemic bias of the information reported in the news articles." [4] For further research, they state that aggregating news across multiple sources would make the data more veritable.
  - The researchers suggested that the embedding model could be improved upon by capturing the sequence of event triggers within the article incorporating event context (i.e. "event triggers [semantically] surrounding a particular trigger") to model the "dependence between specific triggers using an attention mechanism. However, extending the embedding model to capture intra-article dependencies requires greater trust in the verisimilitude of the article's content itself. A direct expansion of their work can carry bias into the IE. In a hypothetical example, an article about "a bear attack" becomes an embedding representation of "deforestation politics." Therefore, a more objective source of data is necessary to accurately embed events in the universe of possible events.
- The second problematic attribute of the REN embedding model relates to how the embedding model reduces events to a single keyword. Figure 2.3 highlights keyword contradiction, where a hypothetical example displays

how two events can be identical to another while having a different sentiment or involved actors, leading to different price swings. The model does not capture these nuances.

Summary of Keyword Structure Representation

- **Embedding Style**: Central Keyword
- **Strength**: Keyword detection is highly accurate (93%)
- **Weakness**: Discards semantics and context hierarchy; model expansion would rely on bias data
- **Key Takeaway**: Chakraborty et al. showed that the LSTM + word2vec embedding model yielded poor performance, owing to noisy representations of entire articles. We observe the need for a framework capable of handling higher-order relations and an operationalizable definition of high-order relevance.

Tuple Structure Representation

Luss and d'Aspremont frame relevance as a classification task applied to bag-of-words (BoW) embedding of articles. For BoW, every word occurrence contributes to the term frequency (TF) of the document. Each article is then labeled according to the next-day stock price:

- Positive label (+1): stock price increased
- Negative label (–1): stock price dropped

Then, Luss and d'Aspremont use a Support Vector Machine (SVM), trained to classify BoW vectors into +1 or -1 labels. The SVM learns a decision boundary (hyperplane) that separates the two classes in the high-dimensional BoW space. This model only performs well when the vocabulary is domain-specific (less sparse vectors). Ding et al. noted the shortcomings of a legacy model from Luss and d'Aspremont:

- BoW treats words as independent features.
  - "Apple sues Samsung" and "Samsung sues Apple" have the same words, so they will have a similar vector representation.
  - Role relations are not modeled.
- The input vectors are extremely spare and high-dimensional, so large datasets are required to avoid overfitting.
- The model can't generalize across paraphrased events because word similarity is not embedded.

Ding et al. propose a "relational database embedding" structuring methodology [6]. Relational database embedding refers to a family of models that learn vector representations of structured data, typically (entity1, relation, entity2). Similar to Chakraborty et al.'s use of a custom embedding for representing the central topic of an article, Ding et al. represent each central event further with a

tuple for the central event (Subject, Verb, Object, Time), extracted only from an article title. A tuple would look like ("Google", "Acquires", "Nest", Jan 13, 2014"). Since roles are position-dependent, the former is not equivalent to ("Nest", "Acquires", "Google", Jan 13, 2014"). This event representation is reminiscent of GDELT's events database structure.

Ding et al. deployed Open Information Extraction (Open IE) onto Reuters and Bloomberg article titles. Open IE is a natural language processing (NLP) technique that extracts structured facts from unstructured text [7]. Since OPEN IE returns many facts, Ding et al. extract only valid (Actor, Verb, Object, Time) tuples. Next, tuples are embedded with a Neural Tensor Network (NTN) into dense embeddings that preserve the semantic roles.

Finally, Ding et al. combine the embeddings with a Convolutional Neural Network (CNN). A major strength of the model architecture is that events are not treated as 'one-off.' Ding et al. enable event dispersion across time with a CNN that learns semantic patterns across recent (either mid- or long-term) events. The CNN's target is a binary class +1 or -1, indicating a prediction for the index to go up (+1) or down (-1). In Figure 2.4, Ding et al. illustrate the effects events can have on stock prices.

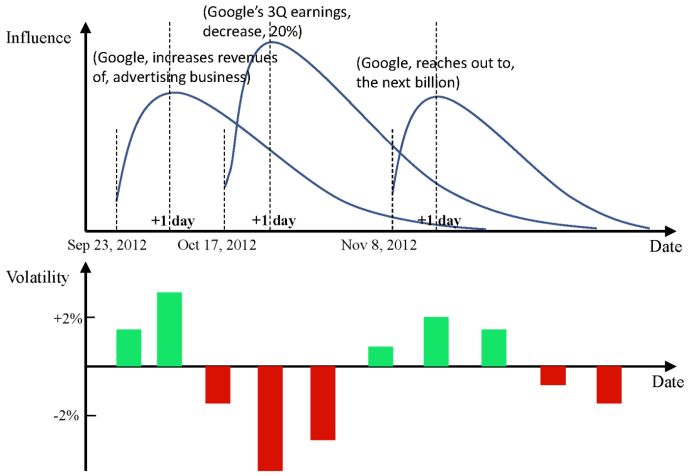

Figure 2.4: Ding et al. show a concrete example where news articles and stock prices are correlated with the same time points.

For mid- and Long-term windows:

1. One vector per day, resulting in a sequence of vectors weekly/monthly.
2. A 1D convolution is applied with a 3-day sliding window (models event co-occurrence and succession).After convolution, max pooling is used to select the most influential temporal features across the window.
3. After convolution, max pooling is used to select the most influential temporal features across the window.

For the short-term window, the CNN is not used. They average event embeddings from the previous day. The outputs of the three time windows are

concatenated into a single vector, $VC = [V_{\text{long\_term}} \mid V_{\text{mid\_term}} \mid V_{\text{short\_term}}]$. Finally, VC is passed through a hidden layer (a non-linear transformation) and a final output layer (sigmoid activation reveals probability of class +1 or -1). The CNN yielded a 64.21% accuracy for Ding et al.

Yet there are some important limitations.

- The model does not predict quantities, only direction.
- The model predicts only the next day. For this paper, we are concerned with long-term forecasting, which is critical for domains beyond stock market prediction (e.g. in the domain of public health or supply chains).
- The embedding pipeline completely reduces semantic complexity.
- The NTN does not learn any relationships between events to model less frequent events.
- Event diversity is limited to what appears in Bloomberg/Reuters titles. The dataset contains no coverage of global social, political, or natural disaster events unless they affect markets directly.
- If the central fact in a news article is "Nvidia didn't reach quarterly earnings," the tuple might look like ("Nvidia" "reach" "earnings"). When the text is reduced to the tuple structure, false information or under-informing information can be embedded.
- The authors make explicit that the goal is to predict stock prices – making this framework domain-specific. Tuple extraction is more simple with natural language from financial news sources, because the domain provides frequent text patterns.

Summary of Tuple Structure Representation

- **Embedding Style**: Central Tuple
- **Strength**: Structures the core proposition of an event, reduces sparsity.
- **Weakness**: Still misses high-order semantics (nuance) and context hierarchy (modifiers), leading to lack of generalizability and more precise prediction beyond directional prediction.
- **Key Takeaway**: Ding et al. concluded that tuples are better covariates than words for stock market prediction. We assert that this tuple scheme evolves Chakraborty et al.'s keyword and can be evolved further to (i) maintain semantic complexity (ii) reduce noise and (iii) potentially generalize across domains.

Summary of Labeled Event Representation (Relational Relevance)

We have shown that operationalizing relational relevance, whether with keywords or tuples, fundamentally misleads prediction. The next step forward is to learn the cross-event structural relations, rather than expressing events internally.

2.2.2 Structuring in High-Dimensional Semantic Space (Centrality Relevance)

This section reviews an implementation of Definition 1.5. We argue that Wikipedia provides a richer source of information closer to the truth. Wikipedia2vec embeddings offer high-order semantic preservation, encoding entity relation (link graph) for directness, and anchor text (words surrounding links) for nuance.

GANs

This section discusses GAN-Event LSTM from Kalifa et al., who proposes an avenue to maintain higher order semantic complexity, verisimilitude and structure [8]. GAN-Event LSTM (1) utilizes the Wikipedia2vec embedding model, potentially avoiding the issue of bias altogether, and at the same time contains the relations of events in semantic space (via Wikipedia link graph) (2) learns a structured representation of events and their relations with a GAN. Wikipedia2vec online demonstration visualizes the embedding space, shown in Figure 2.5.

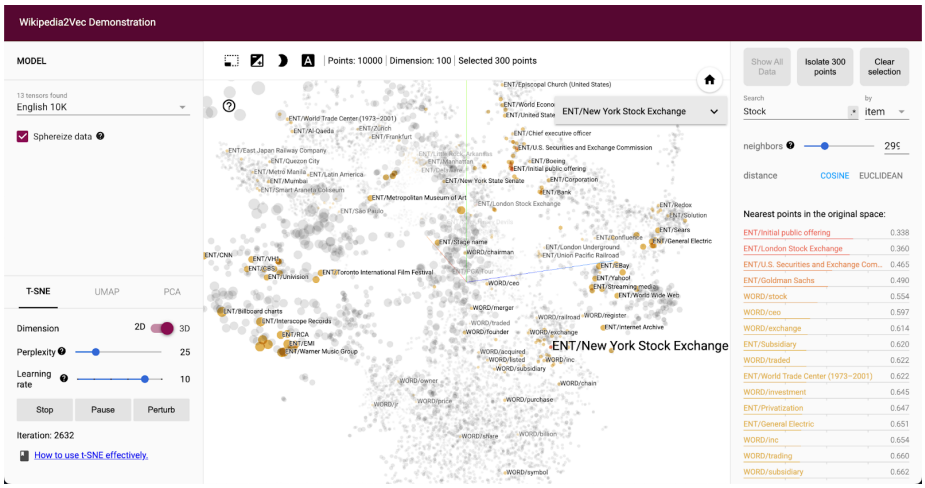

Figure 2.5: Using the Wikipedia2vec online demonstration, we queried for "New York Stock Exchange" to visualize the embedding model's distance-based network.

GAN-Event LSTM shows how integrating wide-ranging world events into e-commerce sales forecasting significantly improves performance during anomalies [8]. Unlike common practice to cherry-pick specific events, the model leverages 20,000 events, dating back to 1980, with 2-4 events per day.

The model uses a transformer-based adversarial encoder that learns the strength of association of events occurring on a single day, thereby filtering out irrelevant events and emphasizing significant ones. The encoder is trained with two objectives: (1) reconstruct the masked set of events that occurred on that day and (2) predict hidden events with the context of unmasked events, which forces the model to learn the relationship between the events.

The idea behind learning the relationship between events is to 'repackage'

those events as a unified, structured signal for an LSTM. The LSTM is trained on daily sales data from eBay along with these daily world event representations; over time, the LSTM learns the relationship between a day's textual representation with the resulting sales of products. This paper demonstrates that when the residual component of a time series is higher – meaning the time series exhibits more anomalous behavior – the GAN-Event LSTM significantly outperforms every other baseline model. For example, when comparing COVID-driven sales surge of Football Cards on eBay, the GAN-Event LSTM achieved 50% lower Mean Absolute Error (MAE) over the five days in 2020 with the highest residual component, as compared to baseline models.

Category	Model	K=5		K=10		K=20	
		MAE	WMAPE	MAE	WMAPE	MAE	WMAPE
Cards	ARIMA	6962	0.869	6867	0.927	5752	0.937
	Prophet	8104	1.012	7529	1.017	6264	1.020
	Neural Prophet	7971	0.995	7233	0.977	6012	0.979
	Event Neural Prophet	7802	0.974	7126	0.962	5944	0.968
	LSTM	6453	0.806	6560	0.886	5478	0.892
	GAN-Event LSTM	3239	0.404	4592	0.620	4012	0.653
Cases	ARIMA	7410	0.845	6175	0.815	5329	0.850
	Prophet	8863	1.011	7699	1.016	6421	1.024
	Neural Prophet	9068	1.034	7731	1.020	6125	0.977
	Event Neural Prophet	8877	1.013	7027	0.927	5650	0.901
	LSTM	7263	0.829	6120	0.808	5219	0.832
	GAN-Event LSTM	5241	0.598	3960	0.522	3562	0.568
Masks	ARIMA	26275	1.002	19551	1.006	14010	1.005
	Prophet	26929	1.027	20261	1.043	14409	1.034
	Neural Prophet	27246	1.039	19424	0.999	14514	1.041
	Event Neural Prophet	26113	0.996	19776	1.018	14136	1.014
	LSTM	26284	1.002	19502	1.003	13972	1.002
	GAN-Event LSTM	22166	0.845	16766	0.863	12732	0.913
Watches	ARIMA	1386	0.868	1245	0.864	1121	0.892
	Prophet	1591	0.997	1491	1.035	1318	1.049
	Neural Prophet	1510	0.946	1397	0.970	1260	1.003
	Event Neural Prophet	1491	0.934	1357	0.941	1210	0.963
	LSTM	1394	0.873	1268	0.880	1102	0.878
	GAN-Event LSTM	1006	0.630	844	0.586	816	0.649
Shoes	ARIMA	13063	1.00	9483	1.007	6351	0.990
	Prophet	13079	1.001	9427	1.001	6427	1.001
	Neural Prophet	13043	0.998	9440	1.003	6238	0.972
	Event Neural Prophet	13021	0.997	9523	1.011	6236	0.972
	LSTM	13044	0.999	9451	1.004	6434	1.003
	GAN-Event LSTM	12851	0.984	9231	0.980	6272	0.977

Table 2.2: Forecasting performance across multiple categories (Cards, Cases, Masks, Watches, Shoes) at 5, 10, and 20 highest residual days in each time series. Asterisks denote baseline models; zeros are placeholders.

GAN-Event LSTM Data Distribution

The distribution of events in Kalifa et al.'s dataset is highly skewed toward a few categories. Since the GAN-Event LSTM is conditioned on both historical and future events, their dataset is naturally limited to events like sports and elections, both of which can be assumed to happen. This is similar to how Neu-

ralProphet uses predetermined holidays [16]. Figure 2.6 shows this skewed event distribution. We argue that while Kalifa et al. mention GAN-Event LSTM's strength under abnormal weather and pandemics, their model is similar to existing models with a frequently occurring event pipeline. The main difference is that GAN-Event LSTM does not fix events to a date of the year.

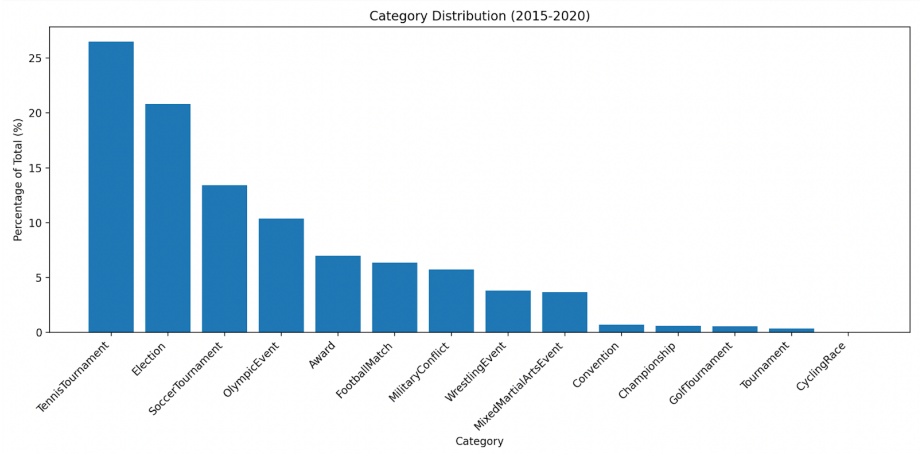

Figure 2.6: The distribution of the 'Category' feature, from Wikidata, for the events dataset used by Kalifa et al. to train the GAN.

This distribution suggests the dataset may be more specialized than intended:

- Elections make up 20% of the category distribution.
- Societal Events make up 20%, which means that Elections comprise 80% of Societal Events.
- With 20% of our data being Elections and 65% being Sports Events, this means 85% of this data is limited to Sports and Elections.
- About 50% of this data is limited to Elections and Tennis.

The motivation behind GAN-Event LSTM is to predict any kind of anomaly, yet the use of future events is restrictive on true anomaly awareness.

GAN-Event LSTM Generalization Potential

Additionally, the financial sector experienced its worst days in history during COVID-19 (the test period for GAN-Event). Testing on an unprecedented event period can mean brittle empirical evaluation; two models which historically performed the same might diverge during anomalies, where one can outperform the other just as easily as vice versa in the next crash. We should be equally critical of the tests conducted in this report, since we are also only testing on the S&P 500 and focusing on the volatility of 2018.

The covariates for the GAN-Event LSTM are effectively a single, 100-dimensional repackaged 'structurally accurate' representation of a day's events. Kalifa et al.

deploy a GAN to learn the most highly associated events of a day through transformer-based reconstruction (i.e. Given events A and B, predict the third event C). These reconstructed events are pooled into a single day-level embedding. Because the generator's goal is to 'fool' the discriminator into believing this embedding represents a real day, it is effectively optimizing for the most structurally realistic geometric center, illustrated in Figure 2.7.

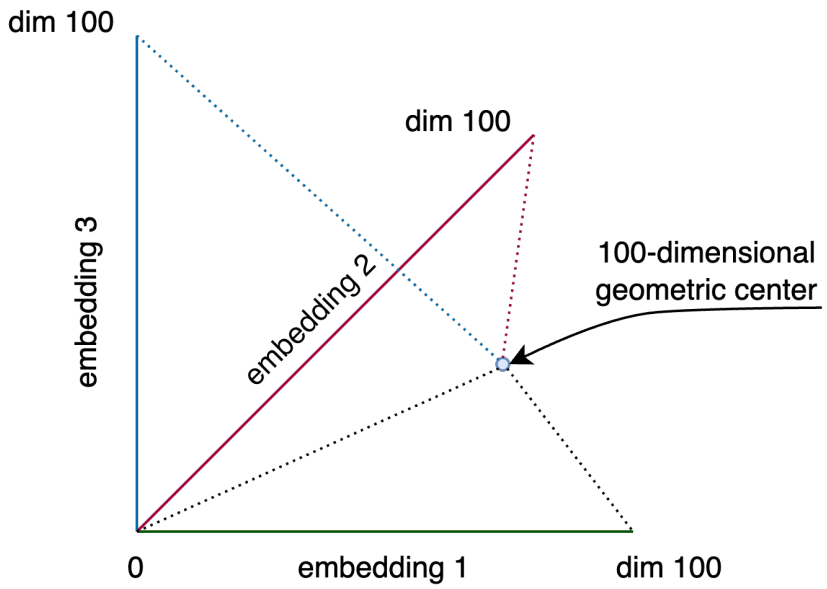

Figure 2.7: Example day-level embedding at the 100-dimensional geometric center of 3 events from a day, obtained by averaging.

We must ask, is this geometric center always informative for prediction? An ideal dataset would contain all world events, and the ideal GAN would learn the relations between them, represented in day-structure. But since we cannot guarantee an ideal dataset, performance depends not on event coherence alone, but on their relevance to the forecasting quantity.

In theory, this approach is powerful–not only reducing the dimensionality of the embedding space but also restructuring the data in a semantically meaningful way–but the LSTM can underperform with event signals that are weakly or not at all correlated to the quantity for prediction.

GAN-Event LSTM was tested only in the COVID-19 anomaly period of 2020. Certain datasets surged in e-commerce product demand, but the S&P 500 experienced a flash crash. Accurate prediction under both dataset depends on how the LSTM learns to associate between events and prices, and this depends on the events dataset. If the events dataset under-explains the system the time series exists in, deep learning predictions can swing unpredictably. This means that one positive performance for 'Masks' in 2020 could be poor in a 2025 prediction timeframe. We should be equally critical of the tests conducted in

this report, since we are also only testing on the S&P 500 during one crash in 2018.

Summary of Structuring in High-Dimensional Semantic Space (Centrality Relevance)

We conclude this section with an appreciation for Relational Relevance under Definition 1.6. The authors cite their goal of handling anomalies like pandemics and abnormal weather, but we hypothesize that the current limitations of GAN-Event LSTM are a lack of event diversity and the choice of datasets highly correlated with the events dataset, limiting generalizability to true anomalies.

2.3 Literature Review Conclusion

The evolution of ideas can be mapped:

Bag-of-Words → keywords → tuples → GAN
Unstructured signals → structured embeddings → structured signals with purpose
Pattern recognition → anomaly handling → meaningful synthesis

The natural language can be unstructured, like with a bag-of-words approach, or it can be structured, as with a finite tuple, keyword, or GAN approach. Below is a map of the research we have done. This is the visual of the connectedness of ideas. For example, (3) does not use TF-IDF but does use a word embedding, and (4) uses categorical features along with an event embedding and a link graph.

The evolution of ideas concludes with the notion that Relations $\neq$ Relevance.

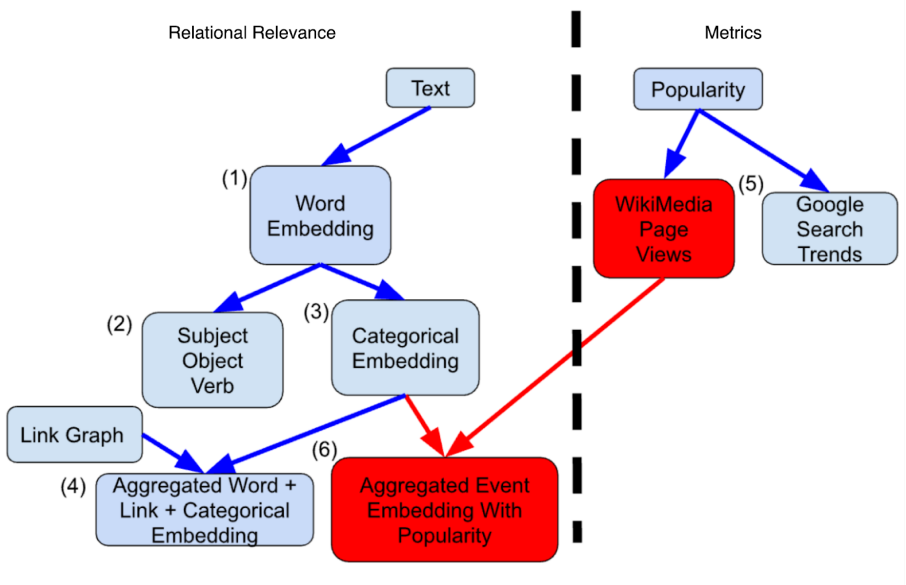

Figure 2.8: TF-IDF bag-of-words [10], (2) Neural Tensor Network [6], (3) central event keyword predictor [4], (4) day-embedding [8], (5) magazine title keywords (6) day-embedding including additional categorical features for popularity.

Figure 2.8 shows one hypothetical addition in the forecasting landscape in box 6: Using popularity metrics as a means to inform the LSTM about the impact of an event across time, in order to sort events by relevance.

Given these concerns, we will explore various notions of defining relevance under new metric-based rules: popularity metrics for temporal relevance diffusion and, semantic metrics for topological meta-distance.

In summary, event-informed models consistently demonstrate superior performance during pattern-breaking anomalies, though they remain vulnerable to noise.

CHAPTER 3

Metrics for Relevance

In Section 3.1 and 3.2 we will show different angles to implement a notion of relevance as structural coherence, inspired by Kalifa et al. These angles can easily be incorporated into the GAN-Event LSTM architecture, but we will dissect whether or not they should be, or if they fall to the same assumptions of GAN-Event LSTM.

3.1 Popularity Metrics for Relational Relevance

While popularity is not a primary component of the TAC-GAN-Event architecture, it plays a crucial role in the conceptual evolution of information relevance that led to the final design. This section will explore different data sources for popularity and how to use them, focusing the discussion around:

- Linguistic ambiguity
- Social/linguistic/economic nuance in metric bias
- Whether popularity is tied to an event's semantic representation at all
- Quantitative modeling of popularity metrics that maintains verisimilitude while restructuring textual data for relevance
- The tension between proxy features and true causality (true relevance)

There is a strong basis for structuring textual data according to popularity metrics. Research conducted by Omar et al. proves there is a tangible link between word popularity, measured by Google Search Trends, and magazine sales prediction [12]. The positive results in Figure 3.1 reinforce existing marketing knowledge that greater consumer interest in article titles leads to more clicks, similar to how Omar et al. transformed magazine title data to predict magazine sales. However, deciding on a single keyword for an event is a more complex task.

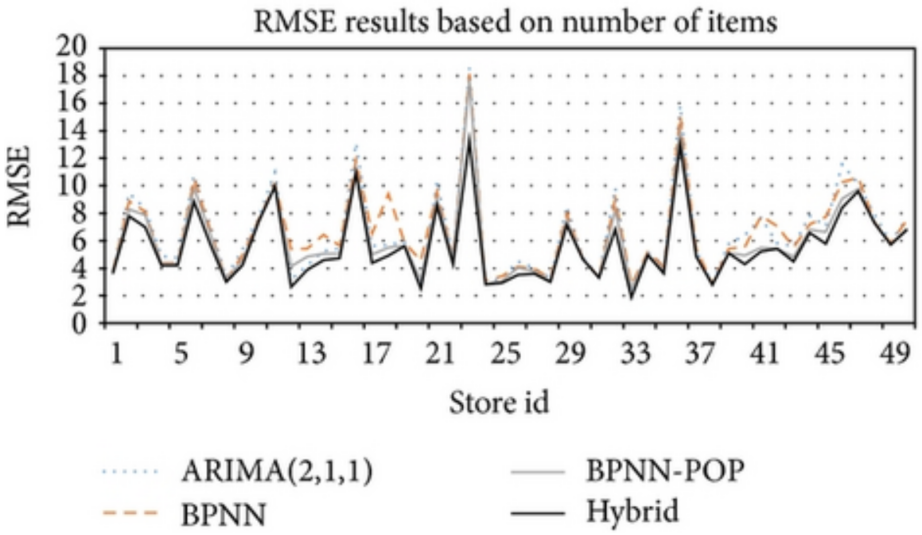

Figure 3.1: The black line is the RMSE for the model utilizing ARIMA and a Backpropagation Neural Network that takes as additional input the popularity of terms [12].

If the link between words and popularity were applied to the problem of structuring event relevancy (under a notion of popularity), we would face some difficulty with Google Trends.

Leveraging Google Search Trends for event relevance is challenging. Query results vary dramatically depending on phrasing and keyword choice. Choosing keywords opens up varying results, depending on the combination of words within the page title from Wikipedia, or even words outside of that name.

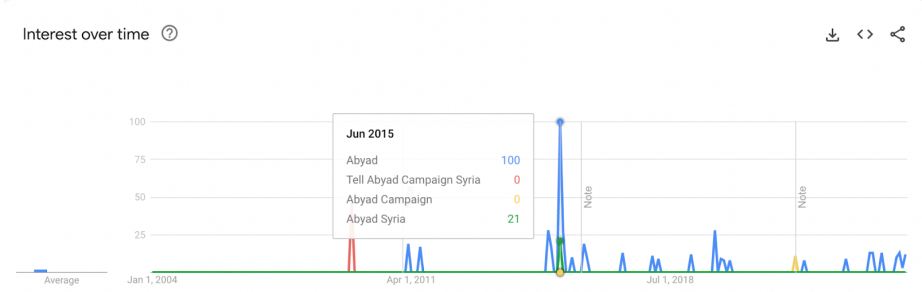

Figure 3.2: Google Search Trends Results for related searches.

For example, Figure X displays the google search trends related to the Wikipedia page “Tell Abyad Campaign Syria”, a military campaign occurring in Searching by the full Wikipedia title “Tell Abyad Campaign Syria” results in zero search interest at the time of the Campaign in June 2015. Querying only “Abyad” results in much more popularity. However, Abyad also means “white

hill" in Arabic, which could explain the fluctuations across time. While the spike likely reflects the conflict, it's unclear which fluctuations represent genuine interest and impact versus unrelated noise, owing to the interconnectedness of words. How do we know which keywords people are searching for when they are interested in the event? This "Tell Abyad Campaign Syria" example highlights the difficulty of using search trends to model real-world interest: Semantic ambiguity distorts signal clarity.

In another example, a simple keyword like "recession" would be insufficient to explain the interconnectedness of simultaneous real world events. As shown in Figure X, even seemingly clear event labels can diverge in their predictive implications, undermining the validity keyword-based embeddings.

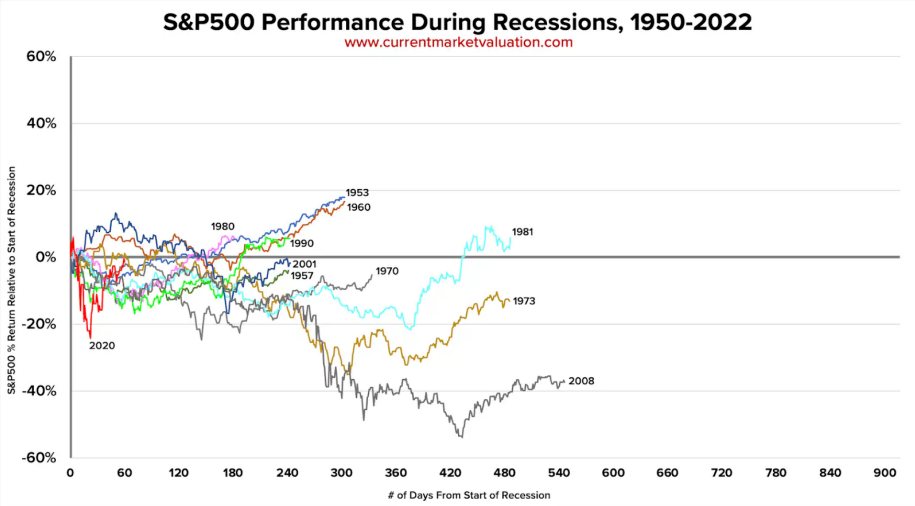

Figure 3.3: Market reactions can vary greatly to the same keyword "recession" [2].

While Google Search trends provide a richer, globally accurate representation of impact, we can rule this out as a feature source to structure event embeddings.

Given the limitations of Google Trends, we turn to Wikipedia as a source of both textual and popularity data, allowing us to study their alignment directly.

We train a linear regression model to take 100-dimensional Wikipedia2vec embeddings and output a predicted 120-day page view sequence. We train the model with parameters:

- **Number of hidden layers**: 10
- **Hidden dimension**: 512
- **Epochs**: 100
- **Learning Rate**: 0.001

The training result is a 0.52 logarithmic MSE. This value comes from the difference (MSE) between the predicted and true values after both have been transformed into the log1p space with:

$$\log 1p(x) = \log(x+1)$$

This transformation handles skewed distributions of page views. To interpret the logarithmic loss, we can convert the error:

$$\text{MSE}_{\log} = 0.52$$

$$\text{RMSE}_{\log} = \sqrt{0.52} \approx 0.721$$

$$e^{0.721} \approx 2.056$$

This means that the model is off by an average factor of about 2.056, indicating moderate performance (likely can be further optimized) in the context of page views that range from 10 to 100,000. Whether this is "sufficient" depends on the use case, but we simply trained the model to ask whether there is a correlation between text and popularity. Since we found a moderate correlation, we have reason to dive into the use case of popularity features, specifically in how these features can be used to structure relevancy in textual embeddings.

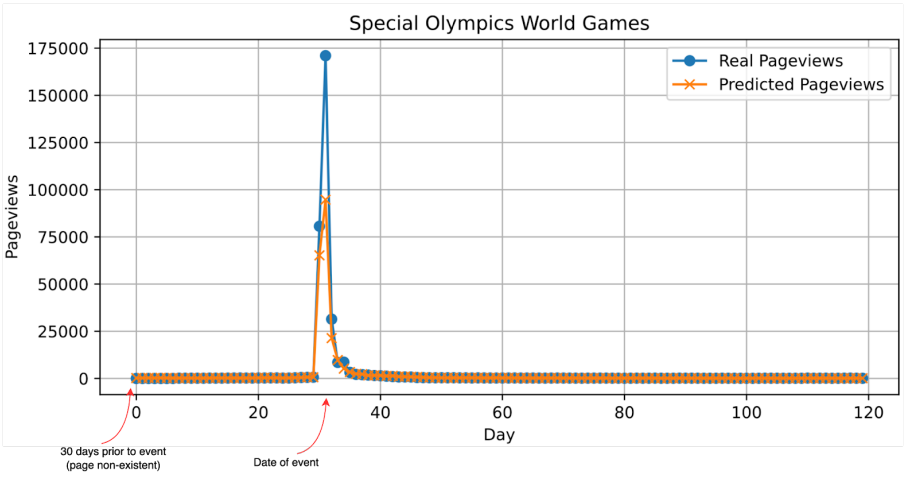

Figure 3.4: A test result from a linear regression model trained to predict 120-day page view sequences with 100-dimensional Wikipedia2vec embeddings.

WikiMedia API provides access to daily page views of Wikipedia pages. This model implicitly assumes a one-to-one correlation between popularity and predictive relevance — an assumption that may not hold uniformly across domains or cultures. In practice, some instances can be devastating in countries which hardly use Wikipedia at all, yet an event in such a country can have a major impact on the quantity we are trying to predict.

To overcome this, there are two approaches: (1) normalize the page popularity of wikipedia pages in the dataset for each country and scale them equally. This yields the z-score, showing how unusual an event is within that country. What follows is a representation of event popularity relative to other events in the same country. This approach avoids underrepresenting major events from countries that don't use Wikipedia as much, showing the impact by deviation

from the normal. (2) Approach 1 assumes that all countries have an equal impact on the quantity we are trying to predict, resulting in biased IE. We can add onto Approach 1 by normalizing all country populations and multiplying the page popularities by those normalized values, thereby defining relevance as a relative value within the discrete finite space of events in a given dataset.

e : an event

c(e) : a country in which an event e takes place

V_e : the raw page views of the Wikipedia page

$\hat{V}_e$: the normalized raw page views of the Wikipedia page

μ_c, σ_c : the mean and standard deviation of page views for events from country c

Pop_c : the population of country c

$\max_c Pop_c$: the maximum population among all countries in the dataset

Approach 1:

$$\hat{V}_e = \frac{V_e - \mu_{c(e)}}{\sigma_{c(e)}}$$

Approach 2:

$$\hat{V}_e = \frac{V_e - \mu_{c(e)}}{\sigma_{c(e)}} \times \frac{Pop_{c(e)}}{\max_c Pop_c}$$

Approach 2 shows how Wikipedia page views can be used as a proxy for impact. But popularity does not necessarily equate to impact, let alone impact on the specific quantity we are trying to predict. Instead, we can rely on the change in popularity, P, within a designated timeframe t. We no longer require popularity to equate to causality – instead, we only care about how interest changes over time. We can classify a page view sequence depending on whether interest is sustained, quickly collapses, or contains more than one sudden spike. As shown in Figure X, the wikipedia page "Special Olympics World Games" exhibits a fast spike and drop.

We have established a solution to dispersing an event's impact over time with a coarse-grained label signal. However, this does not answer the primary questions: "Is this event relevant? What attributes within a particular event embedding are relevant?"

While seeking a measurement for true impact is a futile endeavor, even a more globally and socially accurate metric (z-score normalized popularity) does not equate to a metric for causality, leaving a gap for a better method to structure data for relevance.

3.2 Semantic Similarity Metrics for Relational Relevance

An alternative notion of relational relevance involves enforcing a loss penalty to a GAN's generator for semantic proximity. Rather than restructuring information according to a metric (e.g. popularity metrics), we can define a universal point of maximum relevance semantically.

First, an assumption can be made about the semantic representation of the quantity we are trying to predict. If we are predicting over the "S&P 500" dataset, we can call the Wikipedia2vec function `get_entity_vector()` that gets the embedding of the Wikipedia page with that title. This can be the universal point of maximum relevance.

To incorporate this into the generator's objective, we can leverage the geometry of a trained WIkipedia2vec model's embedding space. This is what we know about Wikipedia2vec embeddings:

- Wikipedia2vec embeddings are trained via a joint loss over word context, anchor context, and link graph [18].
- A Wikipedia2vec model can be trained on a "dump" containing Wikipedia pages, potentially all Wikipedia pages in existence at the time of the dump.
- All embeddings are points in a shared 100-dimensional space.
- Wikipedia2vec situates related concepts near each other via the link graph, so the semantic entity semantic space is structured.

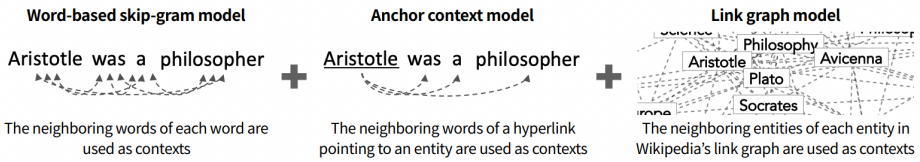

Figure 3.5: Wikipedia2Vec learns embeddings by jointly optimizing word-based skip-gram, anchor context, and link graph models [18].

These context words located next to links, as explained in Figure 3.5, provide additional semantic structure to the embeddings. This loss punishes the generator equally for anchor text and link graph distance, since these are both encoded in the latent space.

The generator's main task remains the same: reconstruct the missing event embeddings to fool the discriminator's decision at the day-level (averaged embeddings occurring in a day). We are not requiring the generator to map to a specific keyword or discrete concept. Instead, we bias its outputs toward a latent zone. This loss can be weighted ($\lambda < 1$) to balance realism and task-alignment.

However, the generator can 'cheat'. The generator can minimize the cosine penalty by reconstructing generic embeddings close to the entity embedding, even if they weren't part of the original embedding set, effectively averaging a few entity-adjacent embeddings, regardless of whether they were in the original event vectors. Even a small loss penalty for entity similarity can override the

generator's main objective. The generator finds a 'shortcut.' To ensure the generator preserves relational integrity, we propose a residual cosine-based loss that penalizes semantic drift. Let us define:

n = number of events in a day
m = number of masked events in a day

e = embedding, dim = dimension in embedding
o = original embeddings
r = original embeddings + reconstructed embeddings
a = the axis; the entity embedding of the forecast-relevant entity

$$o_{\text{avg}} = \left(\sum_{k=1}^{n} \sum_{\text{dim}=0}^{99} e[\text{dim}] \right) \div n$$

$$r_{\text{avg}} = \left(\sum_{k=1}^{n+m} \sum_{\text{dim}=0}^{99} e[\text{dim}] \right) \div (n+m)$$

In the following naive loss function, the generator tries to move the day-embedding toward the entity, potentially hallucinating and discarding real signals. The naive loss is defined as:

$$L_{\cos} = -\cos\_\text{sim}(r_{\text{avg}}, a)$$

If the entity chosen is "Stock Market," and if the original day was already close to "Stock Market", there's no major penalty/incentive. The generator is penalized for semantic drift and rewarded for alignment—only when context supports it. This is a naive albeit feasible implementation of the increasing information relevancy via increasing semantic alignment. Figure 3.6 illustrates this semantic metric-based loss.

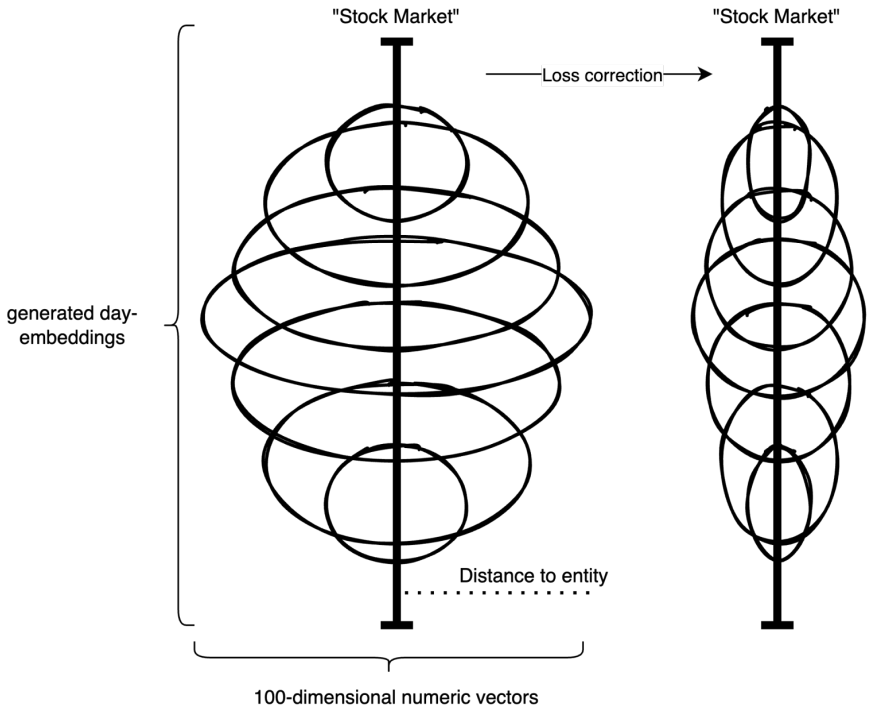

Figure 3.6: A 100-dimensional graph representation of shrinking the distance of day-embeddings from the entity embedding.

We have reviewed the implementation methods involving the extraction of information, resulting in movement towards a maximum point of relevance. This approach relies on the axis choice. Should we use "Stock Market" or "Stock Market Crash" or "Finance"? A perfectly reconstructed event page and the resulting day-embedding will receive directional loss function – even if the generator realistically reconstructs embeddings, and moves them toward the axis, they will never truly reach the axis. Two events that are nearly identical, involving two different politicians, can still exist on opposite sides of the axis. They exist on different sides if one individual is associated with a sports organization, whereas the other is not. This causes a contradiction in the loss function.

3.3 Metrics for Relevance Conclusion

We have reviewed the potential issues of defining a maximum point of relevance. Semantic proximity failed because it collapsed nuance into a static axis. We've shown these two avenues for metric-based Relational Relevance are unreliable. These explorations show that relevance cannot be intrinsic — relevance must be defined by outcome and whether or not the information improves prediction.

- **3.1 Takeaway**: Popularity is a potential avenue for event impact temporal diffusion, but impact $\neq$ popularity. Reliance on Wikipedia page views introduces demographic bias.

- **3.2 Takeaway**: We encounter a semantic contradiction: relative loss function leads to opposition of two events equally similar.

CHAPTER 4

Data Acquisition

This section focuses on data acquisition and engineering. In contrast to Kalifa et al., we (1) introduce a new diversity-focused sampling strategy, (2) rigorously evaluate semantic density through pairwise embedding metrics, and (3) adapt temporal alignment methods for downstream time series forecasting.

We construct a new world events dataset by querying Wikidata and sampling from event categories in a way that mimics the statistical properties of Kalifa et al.'s "world_events_from_1980_dataset" – specifically, its daily frequency and category-level distribution – while correcting for its semantic skew to generalize the model to more anomalies. The dataset will be applied in Section 5.0, where the GAN learns structured relations between events of a day.

4.1 Acquisition of World Events Data

Owing to storage issues, we were not able to download a Wikipedia dump and train our own Wikipedia2Vec embeddings. We were only able to download the pre-trained Wikipedia2Vec embeddings from the April 20th, 2018 dump [13]. Therefore, we will cut off the events data on April 20th, 2018.

To find events data, we used WikiData Query Service [1]. The query example in Figure 4.1 gets all instances of violent crime with the country, date, article name, and category label.

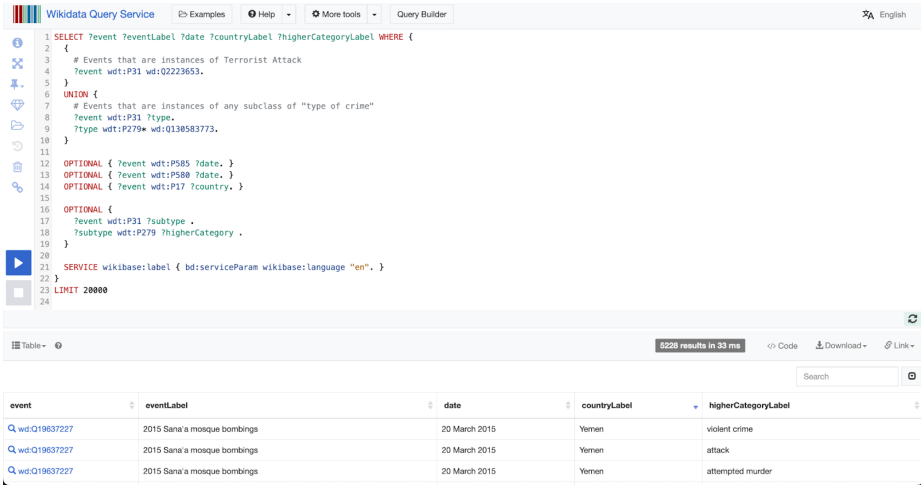

Figure 4.1: Example WikiData query for instances of 'type of crime'.

Wikidata queries can often time out if there are too many subclasses which are also deep. We queried WikiData to get all the subclasses of 'Event' and 'Occurrence' and manually inspected which were useful. Then, we recursively checked these classes to count their subclasses until we found suitable classes to query, avoiding timeouts. The following classes were queried:

- Crime
- Sports
- Conflict
- Natural Disaster
- Convention
- Violence
- Violation of Law
- Disaster
- Court Decision
- Legal Case
- Armed Conflict
- Military Operation

All of the query results were compiled into a single CSV file. We validated that these Wikipedia pages exist with their titles, have a valid country and date. All redundant events resulting from pages belonging to the same class were not included in the final events dataset.

The 'highCategoryLabel' returned from Wikipedia2vec will act as the 'Category' feature in our dataset. We will manually create a dictionary to map

'Category' features to 'High-Category' features – two levels of granularity – just as `wikipedia_events_from_1980` contains [8].


```
"aviation accident": "SocialEvent",
"aviation incident": "SocialEvent",
"battle": "PoliticalEvent",
"behavior": "SocialEvent",
"black market": "CriminalEvent",
"blockade": "PoliticalEvent",
"bomb attack": "CriminalEvent",
"bombardment": "PoliticalEvent",
"brutality": "CriminalEvent",
"by-election": "PoliticalEvent",
"calamity": "SocialEvent",
"calendar date": "SocialEvent",
"capital punishment": "CriminalEvent",
"car bombing": "CriminalEvent",
"cause of death": "SocialEvent",
"change": "SocialEvent",
"cheating": "SocialEvent",
"cheating in sports": "SportsEvent",
"child sexual abuse": "CriminalEvent",
"chronicle": "SocialEvent",
"civil disobedience": "PoliticalEvent",
"clandestine HUMINT": "PoliticalEvent",
"collective memory": "SocialEvent",
"collision": "SocialEvent",
"combat": "PoliticalEvent",
"combined authority mayoral election": "PoliticalEvent",
"combustion": "SocialEvent",
"competition ice climbing": "SportsEvent",
```


Figure 4.2: A snippet of the 320 unique 'Category' values mapping to 4 'High-Category' values.

4.1.1 Wikipedia2vec

We loaded the Wikipedia2vec pre-trained embeddings from the 2018 dump and attempted to get the 100-dimensional embeddings for all pages. Pages which had no embedding were eliminated from the dataset. Lastly, the `wikipedia_events_from_1980` was combined with ours, and all duplicates via the `wiki_name` column were removed.

Our total dataset size becomes 15,091 for all events with embeddings. For the TAC-GAN-Event framework, the events dataset is temporally aligned with the market trading days by shifting ahead to the nearest trading date.

4.1.2 Distribution Matching & Increasing Diversity

With changes to the events data, we have to ensure that the original GAN can still optimize without too much hyperparameter adjustments. By matching key distributional statistics (mean, variance) from the original dataset, we isolate

the effect of semantic diversity on GAN training stability, without introducing statistical noise.

The new dataset, without any redundancy, contains some days with hundreds of events – many of which are still repetitive. For example, local elections in Spain appear dozens of times. We calculated the mean events per day in `wikipedia_events_from_1980` (2.25), as well as the standard deviation (3.54), drawn from a Poisson Distribution. We go about matching these values while maintaining unbiased diversity, as follows:

- `TARGET_CATEGORIES`: We focus on the four "High-Category" values: `[''SocialEvent'', ''PoliticalEvent'', ''CriminalEvent'', ''SportsEvent'']`.
- `global_counts`: Keeps track of how many events of each target have been included.
- `Category Detection`: Identify which `TARGET_CATEGORIES` are present and which are missing in a day.
- `Initial Allocation`: Evenly distribute n samples across all `TARGET_CATEGORIES`. The remainder is distributed round-robin style.
- `Reallocation of missing categories`: If a target category is missing that day, its quota is reallocated to other available categories. Reallocation favors `TARGET_CATEGORIES` with lower global counts to encourage diversity in the dataset.
- `Sampling`: Randomly sample from each `TARGET_CATEGORY` using the allocated quota.

Table 4.1: Comparison of statistical properties between `world_events_from_1980` dataset and our constructed dataset.

Metric	**world_events_from_1980**	**Our Dataset**
1 - Avg. Cosine Similarity	0.6690	0.6136
Cosine Similarity Std. Dev.	0.1380	0.1800
Euclidean Distance Mean	8.4376	5.9071
Euclidean Distance Std. Dev.	1.7323	2.6495
Gini Coefficient	0.2298	0.2492
95th Percentile Euclidean Distance	11.0453	9.7103
High-Cat. Entropy (bits)	1.2737	2.1569
Normalized High-Cat. Entropy	0.8036	0.8344
Cat. Entropy (bits)	3.2022	5.5263
Normalized Cat. Entropy (bits)	0.8411	0.6903
Avg. Events/Day	2.27	1.92
Events/Day Std. Dev.	3.48	3.96
SportsEvent	62.58%	41.78%
PoliticalEvent	26.95%	27.01%
SocietalEvent	10.46%	18.17%
CriminalEvent	N/A	13.04%

The observed metrics suggest that our dataset likely contains multiple semantically dense subregions, or "islands," characterized by high intra-cluster

similarity. This likely stems from our more unique queries. These tightly packed groups skew the overall similarity distribution downward, thereby reducing the global average. In contrast, `wikipedia_events_from_1980` demonstrates a more uniform semantic distribution with embeddings spread more evenly across the space, lacking the pronounced clustering seen in our dataset.

This interpretation is further substantiated by the higher standard deviation and Gini coefficient associated with our dataset's cosine similarity distribution. These values are indicative of greater structural heterogeneity, where small intra-cluster distances coexist with relatively infrequent but large inter-cluster distances. The `wikipedia_events_from_1980`, with lower variance and Gini, appears more homogeneously distributed.

Table 4.2: Comparison of structural observations between datasets.

Structural Observation	**world_events_from_1980**	**Our Dataset**
Intra-cluster distances	Medium	Small
Inter-cluster distances	Medium	Large, infrequent

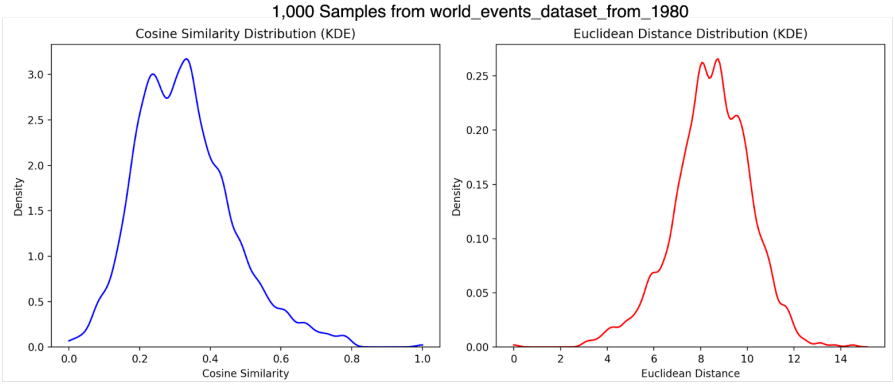

Figure 4.3: Smoothed histograms of 1,000 embedding pairs (out of possible ∼110 million) from `world_events_dataset_from_1980`.

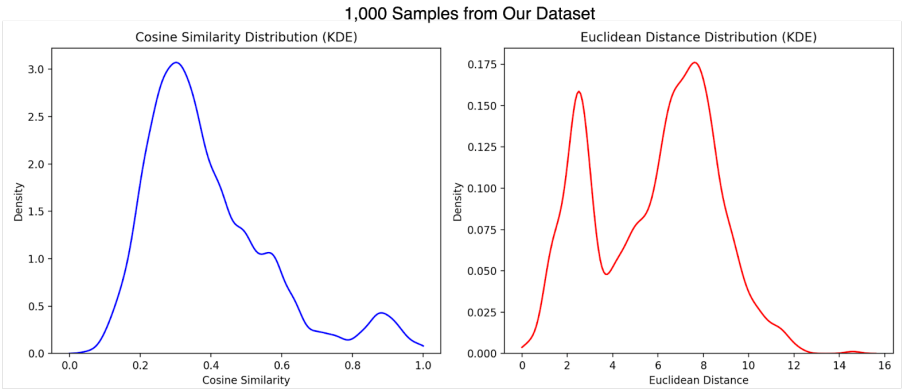

Figure 4.4: Smoothed histograms of 1,000 embedding pairs (out of possible ∼110 million) from our dataset.

The histograms display that even on a small subset of possible pairs, two distinct clusters form in our dataset.

4.2 Data Acquisition for Time Series

We used the *yfinance* library to load past S&P 500 stock data into a CSV file. Stock data is used for time series prediction because it is easily obtainable data. The S&P 500 index was strategically selected for its representation of more than one stock, making it inclusive of more event manifestations.

Figure 4.5: Close Value of S&P 500 from July 1st, 2015 through April 20th, 2018.

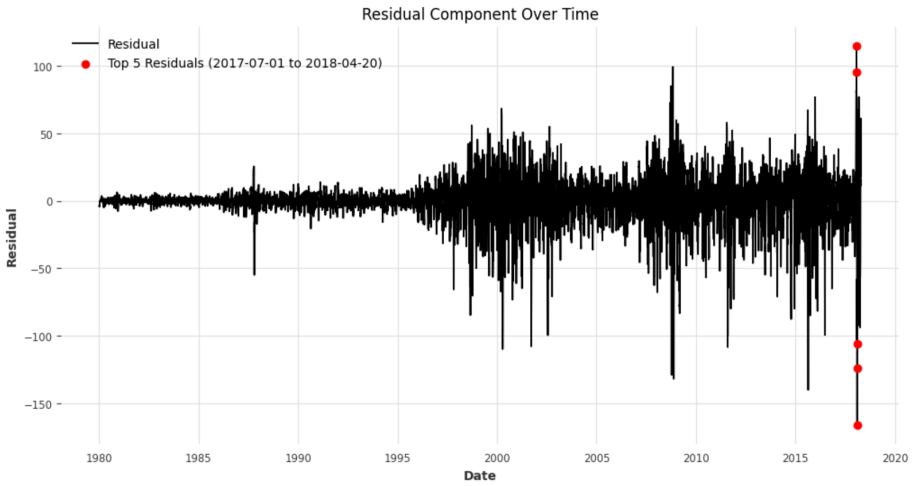

Figure 4.6: Top 5 most anomalous days (highest residuals) occur in our test set, 2017-04-20 to 2018-04-20.

This time series will serve as the forecasting target for TAC-GAN-Event LSTM experimentation. The test set will contain a day with the highest residual in the 38 years of stock data. Strong performance over this dataset is the objective of both GAN architectures.

CHAPTER 5

TAC-GAN-Event Methodological Basis

This section explains the GAN framework, which we implemented with Pytorch Lightning version 1.2.1 and Python 3.7.9.

5.1 GAN-Event

GAN-Event is an adversarial transformer-based encoder framework. Transformers have the benefit of receiving varying input sizes, so the number of daily events is not fixed. Let

$$z \in \mathbb{R}^{n_t \times d}$$

be a vector of event embeddings, where n_t is the number of events on day t, and d is the embedding size. The generator $G(z; \theta_g)$ is a differentiable function

$$G : \mathbb{R}^{n_t \times d} \to \mathbb{R}^{n_t \times d}$$

in the form of a transformer encoder with parameters θ_g. G outputs a vector

$$z' \in \mathbb{R}^{n_t \times d}$$

of generated event embeddings. The discriminator $D(z; \theta_d) \to \mathbb{R}$ is a differentiable function in the form of a transformer encoder with parameters θ_d. D outputs the probability that z came from the dataset, rather than from G, as a single scalar, $D(z)$. G learns to reconstruct events, while D learns the strength of association between the day's events.

Without a GAN, a naive *Event LSTM* architecture simply averages Wikipedia2vec embeddings to get the day-level representation. Kalifa *et al.* show that supplying the raw events to the LSTM (*Event LSTM*) presents a number of pitfalls:

1. **Spurious Correlations**: Since not all events that occur in a day are relevant to each other, some events constitute noise.

2. **Noise**: Pooling all events naively gives equal weights to noise and important information.
3. **Generalization**: *Event LSTM* cannot generalize well to one-time or novel events.

The GAN solves all of these problems by learning an association between events. The Wikipedia2vec model positions similar events closer to each other, creating an angular relational map. Yet when the Wikipedia2vec model was trained on a large 'dump', the events dataset is sparse due to the sheer amount of pages in the dump.

If we understand the GAN architecture correctly, the GAN effectively shrinks the relational map of the Wikipedia2vec distances, learning its own distribution of relations.

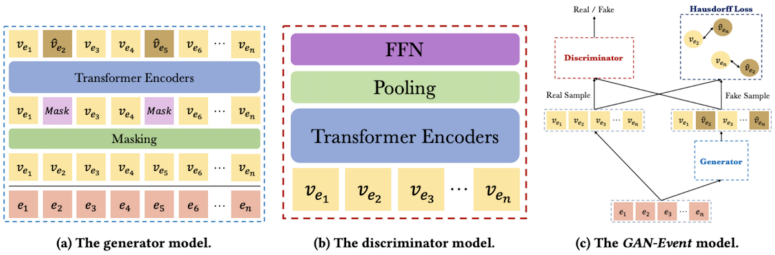

Figure 5.1: A diagram from Kalifa et al. explaining the GAN-Event architecture.

The resulting day-embeddings implicitly encode event dependencies. However, if the generator simply reconstructs the original events, the resulting day embedding may be indistinguishable from a naive average — raising the question: What is the generator actually adding? Perfect reconstruction would mean the pooled GAN output would be no better than just pooling real events. This is why the pooling method is so powerful in an adversarial set-up:

- The discriminator enforces consistency in the reconstructed events.
- The generator, in trying to fool the discriminator, must produce events that 'belong together' in a real day – it cannot generate random embeddings.
- This constraint forces the generator to create a coherent, structured representation of a day's events, which naive pooling (*Event LSTM*) does not do.

5.2 GAN-Event Generator

In formal terms, for every day t in the training set, the generator receives all events in a day as well as the day before it (in an effort to model the delayed

effects of some events). The generator's input is therefore defined as

$$E = E_t \cup E_{t-1} = \{e_t^1, e_t^2, \ldots, e_t^n\}.$$

The events $e \in E$ are represented as the embedding vector v_e from Wikipedia2vec. Let the day's representation be

$$V = \{v_1, v_2, \ldots, v_n\}.$$

The generator randomly masks a fixed percentage $k\%$ of input events, where k is a predefined hyperparameter. Let the set of masked events be

$$E_{\text{mask}} \subset E,$$

and the new vectors are V_{mask}, where $V_{\text{mask}} \in \mathbb{R}^n$. Reconstruction occurs at each generation step when we perform a forward pass through the generator's transformer encoders:

$$\widehat{V} = G(V_{\text{mask}}).$$

The outputs are the generated events:

$$\widehat{V} = \{\widehat{v}_{e_1}, \widehat{v}_{e_2}, \ldots, \widehat{v}_{e_n}\}.$$

Generation quality is measured by the cosine distance between the input vectors and the generated vectors. For each day t, we minimize the following reconstruction loss over the masked events:

$$\min_{\theta_g} L_{\text{rec}} = \sum_{e \in E_{\text{mask}}} \left(1 - \cos(v_e, \widehat{v}_e)\right).$$

5.2.1 Hausdorff Loss

A challenge in evaluating reconstruction accuracy is that a naive distance loss assumes list element order. Kalifa *et al.* point out that the event embeddings should be treated as an unordered set as opposed to an ordered list. They describe a case where the generator is required to generate e_2 in the 5^{th} place and the e_5 in the 2^{nd} place. Even if reconstruction is successful, the generator may put them into a different position than they were in originally. L_{gen} compares events as an ordered list, so the generator will be penalized.

Hausdorff Distance can solve this problem. X and Y are two non-empty subsets of a metric space (M, d), where

$$d(a, B) = \inf\{d(a, b) \mid b \in B\}$$

is a distance function between a point a and a subset B. The Hausdorff distance between X and Y is defined by:

$$d_H(X, Y) = \max\left\{\sup_{x \in X} d(x, Y),\ \sup_{y \in Y} d(X, y)\right\}.$$

This metric measures how far apart two subsets of a metric space are, in unassuming order. Two sets are close in the Hausdorff distance if there exists a matching between the two that minimizes the pair's distances.

5.2.2 Reconstruction Loss

The Hausdorff distance will measure the distance between original event embeddings and generated ones, but by itself it cannot be used as the loss function. To make the Hausdorff distance differentiable for backpropagation, we approximate the infimum and supremum operators with minimum and mean functions respectively, following prior work on soft set comparison. The loss function becomes:

$$\min_{\theta_g} L_{\text{rec}} = \frac{1}{2}\left(\frac{1}{|E_{\text{mask}}|}\sum_{e\in E_{\text{mask}}}\min_{e'\in E_{\text{mask}}} d(v_e, \widehat{v}_{e'}) + \frac{1}{|E_{\text{mask}}|}\sum_{e'\in E_{\text{mask}}}\min_{e\in E_{\text{mask}}} d(\widehat{v}_e, v_{e'})\right)$$

where $d(\cdot)$ is any distance metric set in the hyperparameter. Our experiments only use the cosine distance $d(x, y) = 1 - \cos(x, y)$ since Kalifa *et al.* found this led to the best performance because Wikipedia2vec applied cosine distance in its optimization process.

5.3 GAN-Event Discriminator

Given a set of events in a day, the discriminator differentiates between two scenarios: (1) the events are entirely real, from the dataset, or (2) the generator reconstructed $k\%$ of the events. With

$$V = \{v_1, v_2, \ldots, v_n\},$$

the discriminator's task is to predict 1 (real) or 0 (fake). In the adversarial setup, the generator tries to "fool" the discriminator in the following adversarial loss:

$$\texttt{BCELoss}(D(G(z)), 1) := -\log(D(G(z)))$$

The adversarial loss for the discriminator L_{adv} is therefore:

$$\min_{\theta_g}\max_{\theta_d} L_{\text{adv}} = \mathbb{E}_{v\sim V}\log(D(v)) + \mathbb{E}_{v\sim V_{\text{mask}}}\log(1 - D(G(v)))$$

The final loss of the *GAN-Event* is therefore:

$$\min_{\theta_g}\max_{\theta_d} \lambda_r \cdot L_{\text{rec}} + \lambda_d \cdot L_{\text{adv}}$$

where λ_d, λ_r are parameters of the model, representing the weights of the reconstruction and adversarial losses, respectively.

5.4 Caveats about GAN-Event

First and foremost, the inputs for Kalifa et al.'s LSTM span w days in the past and future, where w is the window size. This means the model is not truly conditioned on unpredicted events. For experimental comparison between our

modified dataset and model architecture, we follow the same procedure. These inputs would be reduced to only the past window in practice.

Encoding dataset-specific relations into the embeddings themselves comes at a cost. This adversarial set-up, based on structural coherence at the day-level, makes the assumption that information is only relevant if it fits into the structure of the day

It seems Kalifa et al. define relevance as 'structural coherence.' While this method maintains high-order semantic relations compared to keyword and tuple methods, it still discards potentially useful information for the downstream task. Coherence can amplify dominant but irrelevant signals, while down-weighting important infrequent ones.

Figure 5.2 shows how the partially reconstructed day-embedding has an altered geometric center in the embedding space. How do we know if relevant embedding information was lost in the shift? How do we know whether the new relations learned, and thereby the geometric center shifts, are relevant to the prediction task and not more noise?

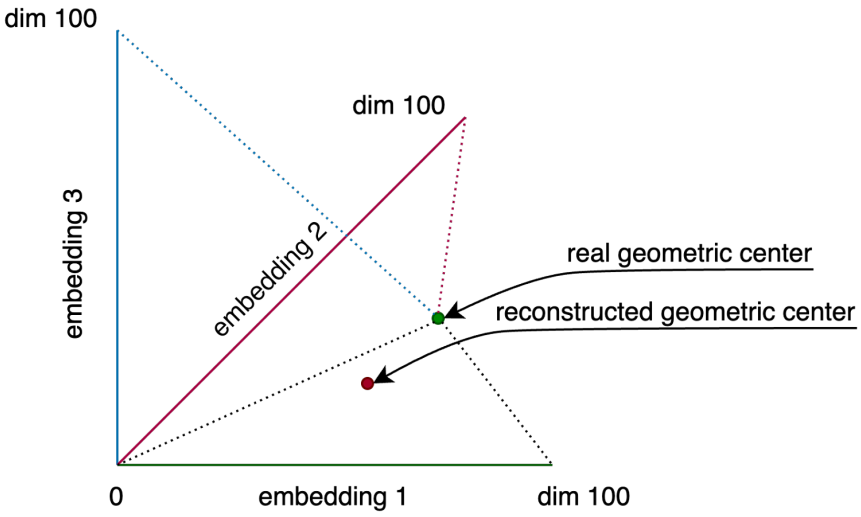

Figure 5.2: An illustration of multiple events in a day with a real geometric center after pooling. The reconstructed geometric center is output from the GAN, which is structurally coherent, and chooses what semantic information is relevant.

This ambiguity motivates our proposed shift: Defining relevance not by relation-oriented structure alone, but by downstream utility.

CHAPTER 6

Utility Relevance Methodology

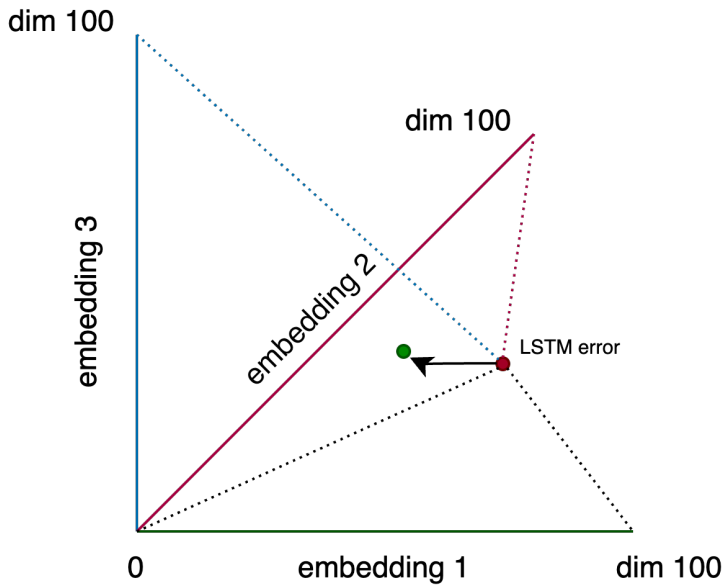

Figure 6.1: The addition of L_{utility} penalizes the generator for representation of semantic noise.

The generator is optimized under a composite objective comprising three competing pressures: (1) local reconstruction via masked event recovery, (2) global semantic coherence enforced by the adversarial loss, and (3) downstream utility, expressed through LSTM forecasting accuracy.

The generated samples become the covariates for the LSTM. To avoid semantic drift, realness and verisimilitude (reconstruction and adversarial loss) is prioritized in the beginning of training. The downstream task loss LSTM is gradually increased each epoch. The LSTM is initialized immediately but only fitted when all days have been generated in training. Then, for each generated batch, we reset the weights of the LSTM and train it again with the new batch of generated days. Performance is then evaluated with wMAPE instead of MAE, because in volatile datasets, absolute scale of the time series varies greatly over time, so wMAPE avoids scaling issues. We can interpret wMAPE regardless of scale.

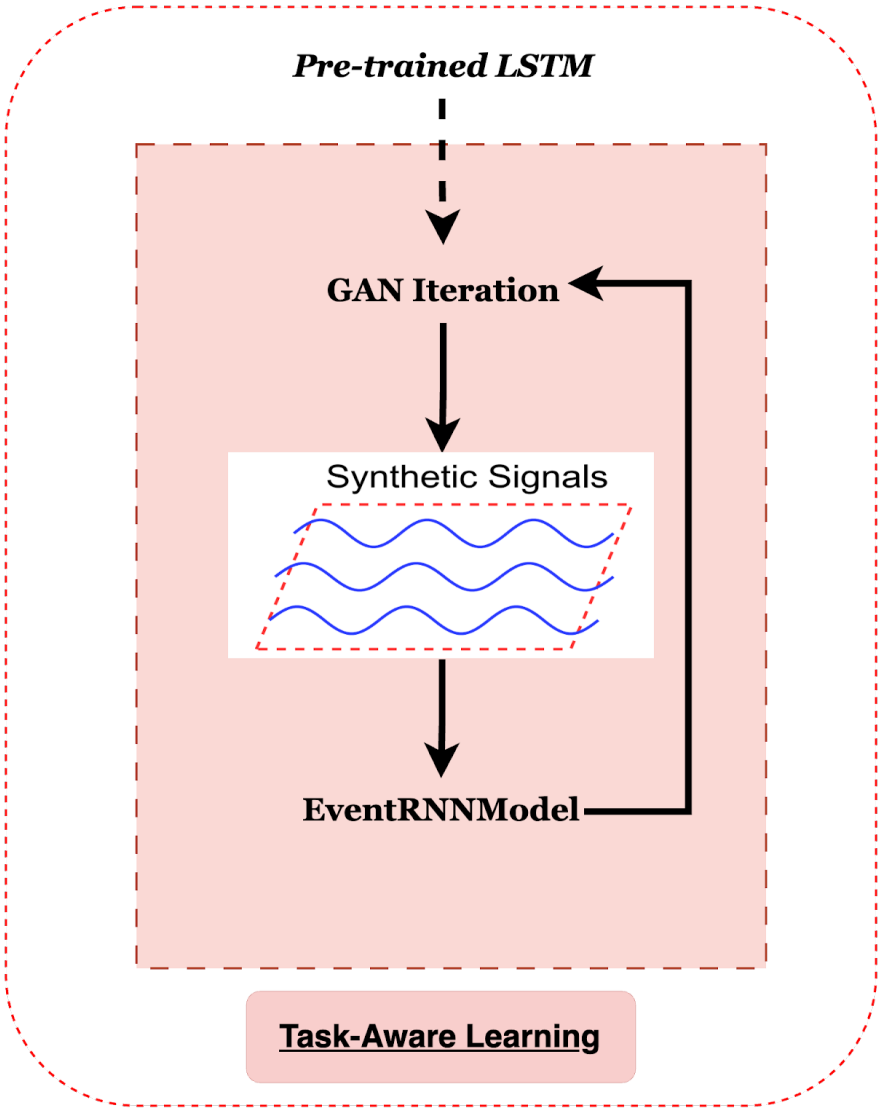

Figure 6.2: TAC-GAN-Event architecture flow.

The discriminator has a much easier task compared to the generator. To prevent the discriminator from dominating in early training, noise and label smoothing are two common techniques [3] [14]. GAN's have been applied to the task of image generation, and there are many well-established architectural designs and diagnostic tools. Since there is not a lot of readily accessible literature on GAN's for Wikipedia2vec embedding generation, we have found that the GAN is hard to stabilize. In fact, the GAN-Event LSTM proposed by Kalifa et al. did not reach convergence. In fact, the training outcome displayed that the generator's data manifold did not have significant overlap with the real data manifold.

We introduce cosine-based input perturbation for discriminator regularization. This technique is specifically adapted to the hyperspherical structure of Wikipedia2Vec embedding space, which is learned using a cosine similarity objective.

Because cosine similarity induces a natural constraint where embeddings are distributed near the surface of a unit hypersphere, we introduce cosine-based perturbations that modify the *direction* of real embeddings without changing their norm. For each real input x, we compute a normalized perturbation η such that $x^T\eta = 0$, and construct the perturbed vector

$$\overline{x} = ||x|| \cdot \left(\frac{x}{||x||} + \varepsilon \cdot \mathbf{n} \right),$$

where ε is a small scalar controlling the perturbation strength. This prevents the discriminator from finding 'hacks' in the distribution, with pressure to learn angular relations.

To ensure that perturbation strength scales meaningfully across different batches and embedding magnitudes, we introduce an `embedding_scale` factor.

This is computed once at the start of training by flattening the batch of real day-embeddings and taking the root-mean-square (RMS) norm across all events of the first batch:

$$\text{embedding scale} = \sqrt{\mathbb{E}[||x||^2]},$$

where the expectation is taken over all events in the batch. Perturbation strength ε is then expressed relative to this `embedding_scale`, ensuring cosine perturbations remain within the embedding hypersphere.

- Early in training (low epochs), stronger noise is applied to real inputs, preventing the discriminator from immediately memorizing fine-grained embedding structures.
- As training progresses (over a window of approximately 40 epochs), the perturbation magnitude decreases linearly to a small minimum, allowing the discriminator to sharpen its decision boundary once the generator has improved.

Specifically, the perturbation strength as a regularization curriculum ε is scheduled as:

$$\varepsilon = \text{embedding scale} \times (0.005 \times (1 - \text{progress}) + 0.0001 \times \text{progress})$$

where:

$$\text{progress} = \frac{\text{epoch number}}{40}$$

This dynamic schedule forces the discriminator to learn generalizable features early on and then specialize later to push the generator harder.

Immediately imposing a large L_{utility} would destabilize training of the GAN and LSTM. We control L_{utility} by scheduling λ_u. We carried out a preliminary hyperparameter sweep on a held-out validation set, testing several utility-weight schedules (e.g., starting weights from 0.1 to 0.3 and linear, exponential, and stepwise increases). We selected the schedule:

$$\lambda_u = 0.2 + 0.25 \times \text{epoch}$$

because it offered the best trade-off between GAN training stability and downstream forecasting accuracy.

This dynamic schedule creates a gentle curriculum for the generator: initially protected from overly confident discriminator gradients, then increasingly challenged as it gains competence on the downstream forecasting task.

The final loss of the *TAC-GAN-Event* is therefore:

$$\min_{\theta_g} \max_{\theta_d} \quad \underbrace{\lambda_r L_{\text{rec}}}_{\text{reconstruction}} + \underbrace{\lambda_d L_{\text{adv}}}_{\text{adversarial}} + \underbrace{\lambda_u L_{\text{utility}}}_{\text{downstream utility}}$$

CHAPTER 7

Evaluation

Our experimentation aims to answer the following questions:

- Can the GAN converge well enough on diverse semantic data?
- Does adding utility feedback materially help downstream forecasting?
- Does utility feedback make covariates more efficient (leaner signals for the same/better performance)?

7.1 Experimental Set Up

All hyperparameters are tunable from run.py (hparams dictionary). Unless stated otherwise, the following were fixed across TAC-GAN-Event experiments:

- `lstm_gradients`: determines whether the LSTM's computation graph remains intact in the feedback loop. A value of False returns a tensor with no gradient history.
- `loss_type`: Hausdorff Loss from Section 5.1.
- `hausdorff_type`: cosine
- Mask $k\%$ of events for reconstruction: $k = 25\%$
- Number of prior days for input: 2
- Batch Size: 32
- Activation: LeakyReLU
- Number of Heads: 4
- Number of Layers: 2

Table 7.1: Training hyperparameters for event embedding models (Test Sets 1–3).

Test Set #	Model	Dataset	Epochs	D LR	G LR	D lambda	G lambda	Weight decay gen	Weight decay disc
Test Set 1	GAN	Kalifa	100	1×10^{-4}	1×10^{-4}	1	10	0.001	0.001
Test Set 2	GAN	new	120	5×10^{-5}	1×10^{-4}	0.5	10	0.001	0.001
Test Set 3	TAC-GAN-Event	new	120	1×10^{-5}	1×10^{-4}	0.5	10	0.001	0.0001

Table 7.2: Training hyperparameters for event embedding models (Test Sets 4–5).

Test Set #	Model	Dataset	Epochs	D LR	G LR	D lambda	G lambda	Weight decay gen	Weight decay disc
Test Set 4	TAC-GAN-Event	new	100	1×10^{-5}	1×10^{-4}	0.5	10	0.001	0.01
Test Set 5	GAN	new	118	1×10^{-5}	1×10^{-4}	0.5	10	0.001	0.01 → 0.001 (epoch 40)

7.2 GAN Convergence and Overlap Analysis

Goodfellow *et al.* were the first to point out that the discriminator is trying to approximate the Bayes classifier between the real and learned data manifolds [**?**]. The best possible discriminator is:

$$D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$$

In our implementation, the discriminator receives an equal number of real and fake samples for each batch (50%/50%). Given this, Goodfellow *et al.* showed that the resulting shared probability mass of the reconstructed data manifold and the real data manifold, called the overlap, is equivalent to $1 - TV$, where

$$TV = 2 \times a - 1$$

where a is total accuracy. We assume a Bayes-optimal discriminator. Under this assumption, these formulae offer only an upper bound on TV and a lower bound on overlap. Therefore, we can approximate the upper bound of the unshared probability distribution mass, as well as the lower bound of the shared distribution.

Key Findings

Table 7.3: Day-embedding Model Convergence

Test Case	Total Accuracy	TV (upper)	Overlap (lower)
Test 1	0.8333333333	0.6666666666	0.333333
Test 2	0.86538461538	0.73076923076	0.26923076924
Test 3	0.704	0.408	0.592
Test 4	0.6905	0.381	0.619
Test 5	0.6465	0.293	0.707

As we can see in Table 7.3, running Kalifa *et al.*'s model without any changes shows that the generator learned at least 33.33% of the real data manifold. When we run the GAN with our more diverse dataset in Test 2, coverage decreases because embeddings are more sparse. At the beginning of training, this

sparsity can cause weak gradient signals because of lower mutual information frequencies between the real and generated distributions, leading to worse performance even with slightly adjusted hyperparameters.

We can't compute $\partial X/\partial i$ for every world event, but under Goodfellow's overlap theorem our discriminator accuracy gives a lower bound on how much of that latent FID our GAN has captured. In future work we can gradient track per-factor contributions from that overlap via 'Category' and 'High-Category' embedding labels. The distribution of the gradient contribution for a category over time evinces the FID of that factor (e.g., earthquake).

Our experiments show an incremental increase in convergence over successive tests. Table 7.3 shows that LSTM performance decreases with covariates whose GAN came closer to convergence. But this pattern breaks between Test 4 and Test 5. In Test 5, the GAN attained the most overlap with the real distribution, but it performed worst overall. Test 4's GAN comes in close second to Test 5's GAN convergence, yet we see a significant difference in performance with the addition of task utility. We hypothesize that an even greater λ_{utility} could have been used in conjunction with other stabilizing hyperparameters to further influence training in the direction of utility.

7.3 Ablative Performance Decomposition: Structural Coherence vs. Task Utility

To show that the feedback loop was not counterproductive, we performed an ablation. During TAC-GAN-Event training, each day-embedding's gradient path was marked prior to LSTM prediction so we can track two different metrics as a time series. The first metric we tracked is the average gradient contribution toward the LSTM prediction. To see contribution stability, the second metric we tracked is the average change in gradient contribution. We also logged the feedback loop's wMAPE, averaged for each epoch.

With these metrics, we track:

(i) how the gradient contribution of events changes over time, depending on if we actually add L_{utility} to the generator's batch-wise loss or not,

(ii) how stable the day-embedding usage is over time,

(iii) how the LSTM performance changes over time based on the wMAPE.

In Test 5, we performed the downstream task the same exact way, and the only difference is that we did not add L_{utility} to the generator's total loss. Figure 7.1 shows that while we observe slower convergence due to the additional training objective, the LSTM performance for TAC-GAN-Event is consistently above its counterpart that does not add L_{utility}.

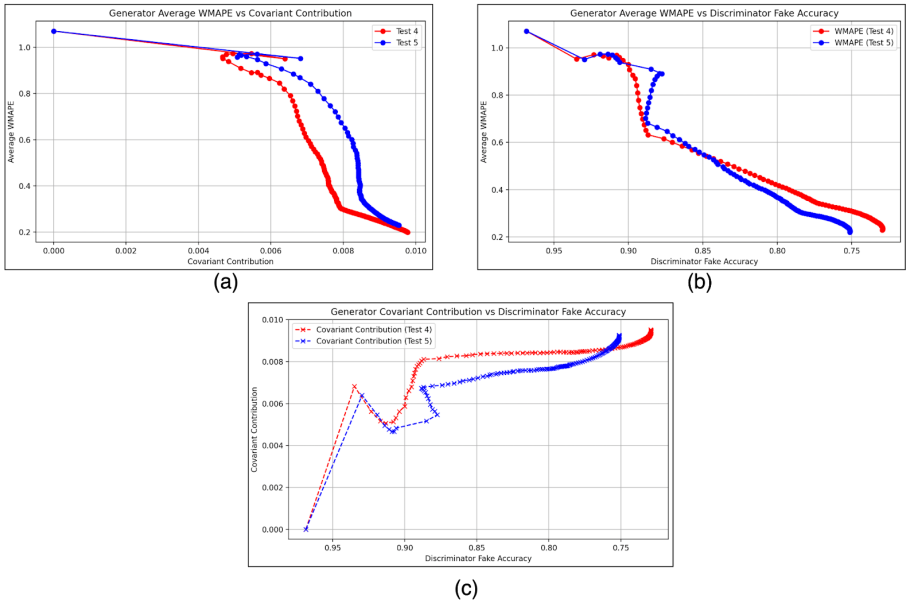

Figure 7.1: Each plot (a), (b), and (c), indexes the y-axis values of a Test to the x-axis. These plots are irrespective of epoch number.

In Figure 7.1, we observe:

(i) Plot (a): At matched levels of covariate contribution, Test 4 consistently achieves lower wMAPE than Test 5 during a stable training phase, confirming that the utility signal improves forecast performance even when semantic input volume is controlled.

(ii) Plot (b): At equivalent discriminator fake accuracies, Test 4 increasingly outperforms Test 5 in wMAPE as convergence improves—suggesting that as the GAN's generator distribution aligns more closely with the true data manifold, utility-driven covariates provide an even greater marginal benefit.

(iii) Plot (c): Despite Test 4's performance edge, the covariate contribution to the LSTM prediction is lower for Test 4 than for Test 5 at matched discriminator accuracies, implying that utility-driven embeddings are more economical: fewer features are needed to deliver higher forecasting gains.

Key Findings

Taken together, these results reveal a clear pattern: as the GAN converges, the day-embeddings produced under the TAC-GAN-Event framework become more precisely aligned with the underlying forecasting task, achieving better performance with leaner, more targeted information. Task-aware covariates are 'economical.'

Our results empirically demonstrate that representations can be optimized not just for structural fidelity, but for task-specific utility, validating the central hypothesis that utility is an optimizable property.

7.3.1 Forecasting Performance

This section applies the learned day-embeddings to the final downstream task. In our case, this is forecasting in 30-day windows over the S&P 500. All LSTMs are trained with dropout=0.3, hidden_layers=404, learning_rate=1×10^{-3}, and 100 epochs. Additionally, the window_size=30, which means the LSTM is conditioned on 30 days of past and future covariates.

Table 7.4: Which prediction model was used.

Test #	Prediction Model
Test 0	LSTM
Test 1	EventRNN
Test 2	EventRNN
Test 3	EventRNN
Test 4	EventRNN
Test 5	EventRNN

Table 7.5: Test results for prediction models on the @K most anomalous days (days with the highest residual values.

Test	Metric	@5	@10	@20	@100	@200	@300
Test 0 (vanilla)	MAE	523.0928	523.0928	523.0928	523.0928	523.0928	523.0928
	wMAPE	0.1964	0.1964	0.1964	0.1964	0.1964	0.1964
Test 1	MAE	232.89	222.46	206.46	158.51	127.18	97.75
	wMAPE	0.0932	0.0905	0.0831	0.0639	0.0509	0.0391
Test 2	MAE	176.30	165.46	150.78	113.18	85.20	63.24
	wMAPE	0.0683	0.0585	0.0585	0.0444	0.0337	0.0253
Test 3	MAE	222.2893	212.8979	202.6437	138.6254	101.0434	76.3243
	wMAPE	0.0849	0.0850	0.0803	0.0553	0.0407	0.0305
Test 4	MAE	235.1569	209.7343	187.2215	121.4637	88.6440	67.2531
	wMAPE	0.0890	0.0780	0.0699	0.0482	0.0355	0.0269
Test 5	MAE	339.2566	320.8467	298.9628	196.2354	142.1670	104.5366
	wMAPE	0.1385	0.1304	0.1206	0.0786	0.0571	0.0420

Key Findings

We interpret the above results:

- In Test 0 the vanilla LSTM (trained without any covariates) completely collapses during the extreme market conditions, but it is superior to all models on a more consistent basis.
- Test 1 had poor GAN convergence, likely capturing only the most prominent patterns. Intuitively, the training for day-embeddings had the worst convergence, implying that the generator learned few meaningful patterns in the data. This means that the Test 1 LSTM received mostly noise with a few frequent and accurate event encodings, which it was able to utilize.
- Test 2 shows the best performance. Test 2 was trained on an augmented version of the dataset from Test 1, allowing the generator to learn more of the same region of the embedding space. Consequently, the LSTM learns

the relation between a price point and with lesser error when the occurrence is more frequent. This is why Test 2 with our dataset outperformed Test 1.

- Test 3 shows the better performance of the two tests that used TAC-GAN-Event day-embeddings. In TAC-GAN-Event training, the generator managed to cover more of the real distribution. As a result, more clear event signals were given to the final LSTM, leading to more complex input. The LSTM did not handle this complex input well, meaning the TAC-GAN-Event day-embeddings were not fully-utility optimized.
- Test 4 and 5 show the value of TAC-GAN-Event. The only difference between them is whether the generator incurs L_{utility}. While the model is faster to cover more of the distribution space, it performs roughly 35.7% worse than task-aware covariates.

7.3.2 Covariate Economy Conjecture

Conjecture 7.3.1 (Covariate Economy)**.** *Let*

$$C = \textit{GAN convergence metric (e.g. overlap lower-bound)},$$
$$G = \frac{1}{N}\sum_{i=1}^{N}\left\|\nabla_{h_i} L_{\text{task}}\right\|,$$
$$P = \textit{wMAPE of the downstream forecast.}$$

Then for any fixed C_0, if two models α (with utility) and β (without utility) satisfy

$$C(\alpha) = C(\beta) = C_0,$$

it follows that

$$G(\alpha) < G(\beta),$$
$$P(\alpha) < P(\beta).$$

In other words, at the same level of GAN convergence, adding the utility term yields strictly leaner covariate gradients and strictly better forecasting accuracy.

We note that while we give an informal derivation (see Appendix X), a fully rigorous proof under non-convex GAN training remains future work. Empirically, however, our paired tests at matched C (Tests 4 vs 5) and conditional regressions in Section 7.4 strongly support this conjecture.

7.4 Summary of Results

Our experiments reveal that structural coherence alone does not guarantee optimal downstream performance under volatility. Injecting task utility into the representation process consistently yields leaner, more predictive covariates. This validates utility-driven relevance as a powerful optimization axis for forecasting under anomalous conditions.

CHAPTER 8

Conclusion

GAN-Event lays the foundation of semantically coherent and embeddings obeying verisimilitude. The evaluation of our model provides early but compelling evidence that task-aware representations offer a promising direction for forecasting models operating under volatility. The ablative comparisons isolate the contribution of task utility under shared conditions. The juxtaposition shows the merit of our definition of relevance as utility, operationalized with TAC innovation.

Limitations and Future Work

There are a few limitations of this work. For one, our stress-test of the TAC-GAN-Event architecture involved only the S&P 500 with a focus on the 2018 anomaly period. TAC-GAN-Event is certainly not a full-proof model and should be tested on other datasets.

The experimentation is a causal setup, not for realistic predictions, because we are isolating new GAN behavior for comparisons to GAN-Event LSTM's setup. To evaluate realistic performance, future work would simply apply past covariates only for the LSTM, ensuring no events are used from the future.

Although we observed consistently stable adversarial training, additional GAN stabilization techniques can be applied beyond our custom angular noise regularizer, such as spectral normalizationa and gradient penalties [11, 17].

Additionally, convergence on highly diverse event embeddings remains an open challenge, inviting new stabilization strategies beyond those explored here. Finally, we face a real-time deployment barrier. Our model is trained on a Wikipedia2vec 2018 dump. If the end goal is a daily forecast pipeline, the utility-focused IE relies on these sequential operations:

1. Timely Wikipedia entry.

2. Retraining the Wikipedia2vec model. Pointing the Wikipedia2vec model at a brand-new page title, it returns nothing from its look-up table.

3. Retraining the TAC-GAN-Event updated Wikipedia2vec embeddings.

Bibliography

[1] `https://query.wikidata.org/`. [Accessed 27-04-2025].

[2] Market Performance During Recessions — currentmarketvaluation.com. `https://www.currentmarketvaluation.com/posts/stock-market-performance-during-recessions.php`. [Accessed 27-04-2025].

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[4] S. Chakraborty, S. Jagabathula, L. Subramanian, and A. Venkataraman. Frontiers in operations: News event-driven forecasting of commodity prices. *Manufacturing & Service Operations Management*, 26(4):1286–1305, 2024.

[5] A. Das, W. Kong, R. Sen, and Y. Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

[6] X. Ding, Y. Zhang, T. Liu, and J. Duan. Deep learning for event-driven stock prediction. In *Ijcai*, volume 15, pages 2327–2333, 2015.

[7] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.

[8] D. Kalifa, U. Singer, I. Guy, G. D. Rosin, and K. Radinsky. Leveraging world events to predict e-commerce consumer demand under anomaly. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, page 430–438. ACM, Feb. 2022.

[9] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[10] R. Luss and A. D'Aspremont. Predicting abnormal returns from news using text classification. *Quantitative Finance*, 15(6):999–1012, June 2015.

[11] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks, 2018.

[12] H. Omar, V. H. Hoang, and D.-R. Liu. A hybrid neural network model for sales forecasting based on arima and search popularity of article titles. *Computational Intelligence and Neuroscience*, 2016(1):9656453, 2016.

[13] S. Ousia. Pretrained Embeddings - Wikipedia2Vec — wikipedia2vec.github.io. `https://wikipedia2vec.github.io/wikipedia2vec/pretrained/`. [Accessed 27-04-2025].

[14] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[15] S. Siami-Namini and A. S. Namin. Forecasting economics and financial time series: Arima vs. lstm, 2018.

[16] O. Triebe, H. Hewamalage, P. Pilyugina, N. Laptev, C. Bergmeir, and R. Rajagopal. Neuralprophet: Explainable forecasting at scale. *arXiv preprint arXiv:2111.15397*, 2021.

[17] T. Xia. Penalty gradient normalization for generative adversarial networks, 2023.

[18] I. Yamada, A. Asai, J. Sakuma, H. Shindo, H. Takeda, Y. Takefuji, and Y. Matsumoto. Wikipedia2vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from wikipedia. *arXiv preprint arXiv:1812.06280*, 2018.

8.0.1 Software environment

Codebase can be accessed at the public repository: `https://github.com/KSkert/TAC-GAN-Event`.

Experiments were run on an M1 Silicone CPU using Python 3.9 / PyTorch 2.1. For training TAC-GAN Event, create a virtual environment `gan` and run:

```
pip install requirements.txt
```

Set the hparams in run.py and run from GAN-Event, run:

```
python -m src.run
```

For Jupyter Notebook experimentation with time series predictions, create a separate environment `ts` and run:

```
pip install requirements_pred.txt
```

Licensing and attribution. This project uses data derived from Wikimedia projects (Wikipedia and Wikidata). These sources are licensed under the Creative Commons Attribution–ShareAlike 4.0 International (CC-BY-SA 4.0) and, where applicable, the GNU Free Documentation Licence 1.3 (GFDL).

In accordance with the ShareAlike condition, all derived data (e.g., embeddings, structured event files) and code in this project are released under the same CC-BY-SA 4.0 licence. Attribution is provided by linking to the source dumps and APIs used, and by including licence texts with the archived artefacts.

The Wikipedia and Wikidata data were retrieved using the following sources:

- Wikidata SPARQL endpoint: `https://query.wikidata.org`
- Wikipedia dump used for embeddings: `https://dumps.wikimedia.org/backup-index.html`
- Wikipedia2Vec pretrained embeddings: `https://wikipedia2vec.github.io/wikipedia2vec/pretrained/`