RevUp: Revise and Update Information Bottleneck for Event Representation

Mehdi Rezaee **Francis Ferraro**
Department of Computer Science
University of Maryland Baltimore County
Baltimore, MD 21250 USA
{rezaee1, ferraro}@umbc.edu

Abstract

The existence of external ("side") semantic knowledge has been shown to result in more expressive computational event models. To enable the use of side information that may be noisy or missing, we propose a semi-supervised information bottleneck-based discrete latent variable model. We reparameterize the model's discrete variables with auxiliary continuous latent variables and a light-weight hierarchical structure. Our model is learned to minimize the mutual information between the observed data and optional side knowledge that is not already captured by the new, auxiliary variables. We theoretically show that our approach generalizes past approaches, and perform an empirical case study of our approach on event modeling. We corroborate our theoretical results with strong empirical experiments, showing that the proposed method outperforms previous proposed approaches on multiple datasets.

1 Introduction

In this work, we are interested in addressing limitations in how computational event modeling can make use of relevant, supplementary semantic knowledge. This is because when modeling text descriptions of complex situations, such as newspaper descriptions of real world events, learning how to encode richer information about those descriptions can be a fruitful way of improving modeling performance (Judea and Strube, 2015; Xia et al., 2021). E.g., if we are dealing with sequences of events, like a newspaper report of a stock or commerce transaction, then being able to encode that "buying" or "selling," even though they may have nuanced connotation differences, are instances of the same general event (a TRANSACTION event) can improve downstream predictive performance on what events might happen next (Ferraro and Van Durme, 2016; Rezaee and Ferraro, 2021).

However, there is not an obvious single way to learn to encode this richer information, as three different questions naturally come to mind: (1) If the model is representationally limited, can we address these representational limitations of the model itself, such as through developing richer latent representations $\mathbf{z}$ of the input $\mathbf{x}$? (2) If there is available background or side information $\mathbf{t}$ that may be especially relevant for the modeling task at hand, can we develop systems that make use of it? (3) Even when side information is available, it may be noisy, e.g., it may not always be available (missing data) or it may contain errors: how can we make our models robust to this noisy side information? In the context of a text-based sequence modeling problem, we propose an approach that addresses all three of these questions.

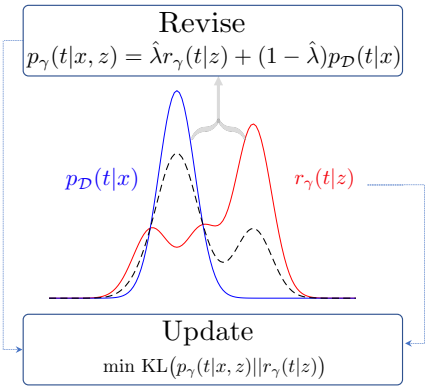


Figure 1: Revise and Update steps for an observed node. We update the proposed distribution $r_\gamma(t|z)$ with the empirical distribution $p_\mathcal{D}(t|x)$ to produce the revised distribution $p_\gamma(t|x,z)$ (dashed). We then minimize the KL divergence between the proposed and revised distributions to update the proposed distribution. For unsupervised nodes, we rely on $r_\gamma(t|z)$ without updating.

We provide a conceptual overview of our method—RevUp—in Fig. 1. The building blocks of RevUp are **revision** of modeling side information $\mathbf{t}$: forming a new distribution by combining the (red) learned proposal distribution with (blue) empirical information about when particular aspects of side knowledge appeared in training, and **updat-**

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 797–814
May 2-6, 2023

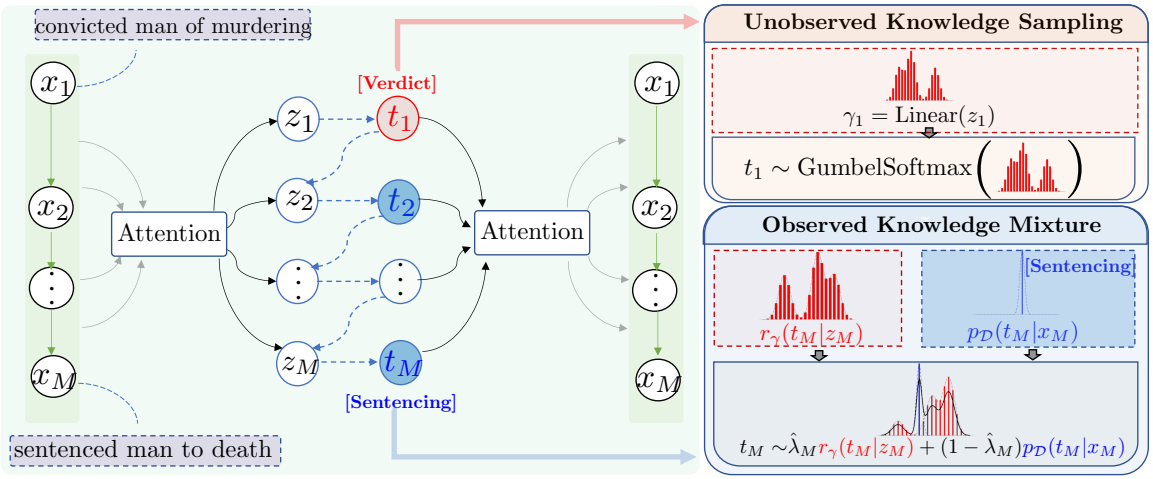


Figure 2: Components of RevUp for event modeling. We encode the sequence of events $\mathbf{x}$ into Gaussian latent $\mathbf{z}$ and discrete knowledge $\mathbf{t}$. Red nodes such as t_1 are latent: [VERDICT] for the *convicted man of murdering* event. RevUp predicts these latent nodes by sampling from the GumbelSoftmax distribution. Blue nodes such as t_M are observed: [SENTENCING] for the *sentenced man to death* event. We use this observed node to modify the proposed distribution and then we draw a sample from the revised distribution.

ing by minimizing the KL-divergence between the new distribution with the previous proposed one.

Our approach is inspired by recent work (Kong et al., 2019), which argued that a key to the success of neural advances in NLP (Mikolov et al., 2013; Devlin et al., 2018; Yang et al., 2019, i.a.,) is that they fall within the information theoretic-based InfoMax framework (Linsker, 1988). In this viewpoint, neural advances can be attributed to implicitly maximizing the mutual information between different representations of the same document.

However, we additionally want to use "side" or "external" knowledge to aid our modeling. The use of side information has long been desired for effective representation (Wyner, 1975; Wyner and Ziv, 1976). While this side information can occur in a variety of forms (Chen et al., 2020; Padia et al., 2018), we broadly view it as a *concise, abstract view of information* conveyed in the main input.

Our work centers on the idea that the latent neural representations $\mathbf{z}$ and side knowledge $\mathbf{t}$ act as complementary representations of the same input $\mathbf{x}$: $\mathbf{z}$ provides a compact representation of the input itself, while $\mathbf{t}$ provides more generic information about the data or task. As such, we use an InfoMax-inspired formulation to incorporate external knowledge into the modeling and latent representation aspects. Since recent work (Joy et al., 2022; Chen et al., 2019) has suggested that providing some sort of "guidance" to these latent variables is beneficial and can lead to these guided models outperforming both fully supervised and fully unsupervised methods, learning with less-than-full observation of structured side knowledge is a requirement. We illustrate this in Fig. 2, where the correct value for the first piece of side information (t_1 = Verdict) is missing but the last piece of side information is known (t_M = Sentencing). Our model must be able to operate over this partially observed sequence of side knowledge as a way of modeling the original event description.

We propose a unified approach, RevUp (**Rev**ise and **Up**date Information Bottleneck), that maximizes the mutual information between the external knowledge and latent representation $\mathcal{I}(\mathbf{z};\mathbf{t})$. We consider the external knowledge $\mathbf{t}$ to be a discrete random variable in a lightly structured, semi-supervised setting. We demonstrate the effectiveness of this approach on multiple modeling tasks. Our contributions are:

(1) We provide a principled, information theoretic approach for injecting side information into neural discrete latent variable models. We provide theoretical backing to show that our methodology captures available information from external knowledge.

(2) We define a new model that leads to state-of-the-art results in the semi-supervised setting on two standard event modeling datasets.

(3) We show that our proposed model generalizes the existing state-of-the-art model studied in Rezaee and Ferraro (2021), where the "parameter injection" method developed there can be

understood as a special case of our framework.

(4) We experimentally show that our model is more robust when the external knowledge is noisy and it outperforms other baselines when the external knowledge is partially observed.

Our code is available at https://github.com/mmrezaee/RevUp.

2 Background

In this section we present the related work and discuss the connections to our approach.

Mutual Information (MI) The mutual information $\mathcal{I}(x;y)$ measures how much two random variables x and y depend on each other. It is defined as $\mathcal{I}(x;y) = \mathbb{E}_{p(x,y)} \log \frac{p(x,y)}{p(x)p(y)}$, where $\mathcal{I}(x;y) = 0$, when x and y are independent. Conditional MI $\mathcal{I}(x;y|z)$ extends MI to measure the conditional dependence between x and y given z, as $\mathcal{I}(x;y|z) = \mathbb{E}_{p(x,y,z)} \log \frac{p(x,y|z)}{p(x|z)p(y|z)}$. When $\mathcal{I}(x;y|z) = 0$, there isn't any information between x and y that is not present in z.

Information Bottleneck Principle (IB) The Information Bottleneck principle is a method for finding the most informative encoding of an input x with respect to the target output y. This is accomplished by finding the maximally compressed encoding z of x that is most informative about y (Tishby et al., 1999). The objective is to maximize

$$\mathcal{I}(z;y) - \beta\mathcal{I}(x;z), \tag{1}$$

where $\mathcal{I}(z;y)$ denotes z and y mutual information, and $\mathcal{I}(x;z)$ denotes x and z mutual information. β balances compression and informativeness.

Variational Autoencoder (VAE) The Variational Autoencoder is similar to the Information Bottleneck Principle in that it finds an encoding z of x to reconstruct x. However, the approach is different as the aim is to approximate the intractable posterior distribution $p_\theta(z|x)$ with a variational distribution $q_\phi(z|x)$ in order to optimize the Evidence Lower Bound (ELBO) (Kingma and Welling, 2013).

Incorporation of Side Knowledge Conceptually, we consider an "event" to be a condensed form of knowledge that outlines part of a particular situation. Semantic frames are a prime example of relevant side knowledge: a semantic frame can be thought of as an abstraction over highly related events. Though they also provide abstractions over the *who*s, *what*s, *where*s and *how*s of events, in this setting, it is sufficient to consider an event's semantic frame as a type of label or cluster id. Multiple potential sources of semantic frames exists, e.g., FrameNet, PropBank, or VerbNet.

Recently, incorporating external knowledge has been investigated by numerous studies in a wide range of tasks beyond NLP (Kang et al., 2017; Flajolet and Jaillet, 2017; Zhang et al., 2021), and zero-shot classification (Badirli et al., 2021). Common across these efforts is treating the side information as part of the input to be encoded. This makes the side information prerequisite knowledge for the model to be learned, rather than supplementary.

A recent approach, called Sequential, Semi-supervised Discrete Variational Autoencoder (Rezaee and Ferraro, 2021, SSDVAE), is a new method for structured semi-supervised modeling that allows, but does not require, side information to guide the learning in an approach called "parameter injection." Because the SSDVAE framework is a deep latent variable model that is specifically designed to treat external knowledge as supplementary to the main task, we focus our study within it and the associated NLP-based computational event modeling tasks examined. They define parameter injection as

Definition. *(Parameter injection) Let* $\mathbf{t} \sim \textit{GumbelSoftmax}(\mathbf{t};\gamma)$*, where* $\boldsymbol{\gamma}$ *are the logits. If* $\mathbf{t}$ *is observed as external knowledge and represented with a one-hot vector* $\mathbb{1}(\mathbf{t})$ *with* $t_{k^*} = 1$*, the operation* $\boldsymbol{\gamma} = \boldsymbol{\gamma} + (\|\boldsymbol{\gamma}\| * \mathbb{1}(\mathbf{t}))$ *guides the latent variable* $\mathbf{t}$ *during the training because on average increases the* t_{k^*} *value (Maddison et al., 2016).*

In RevUp, we build on and generalize this notion of parameter injection, in part by introducing the empirical distribution $p_{\mathcal{D}}(\mathbf{t}|\mathbf{x})$ to accommodate the dependence between data $\mathbf{x}$ and knowledge $\mathbf{t}$.

3 RevUp: Revise and Update

Previous work (Rezaee and Ferraro, 2021, SSDVAE) has empirically shown that incorporating external knowledge in discrete GumbelSoftmax parameters improves model performance with different metrics such as classification, event modeling and training speed. We seek to re-frame this view with information theory terms and generalize it. For fairness and consistency we stick with the same types of computational event modeling tasks that

SSDVAE was developed for. We first describe the probabilistic model we develop (section 3.2) and the loss/training methodology (section 3.3). These enable smoothly incorporating side knowledge into a probabilistic neural model. Recall we provide a conceptual overview of RevUp in Fig. 1.

3.1 Setup

We assume that within a document, we have a sequence of M different predicate/argument-style events, $\mathbf{x} = x_1 ... x_M$, with each x_m describing some action (the predicate) occurring among participants of that action (the arguments). If each event x_m could also be paired with a semantic frame t_m, so we have a corresponding sequence $\mathbf{t}$, then considered across multiple events, $\mathbf{t}$ could provide a sequential generalization. E.g., in the example event sequence from Fig. 2 that includes "convicted man of murdering," and "sentenced man to death," if each event can be associated with a semantic frame (such as VERDICT and SENTENCING for these two events), then the corresponding sequence of frames provides both an abstraction over the entire event sequence, and an incredibly rich, yet low-dimensional, collection of side knowledge.

Having paired sequences $\mathbf{x}$ and $\mathbf{t}$ is not restrictive. Whether $\mathbf{t}$ is partially observed (i.e., not all x_m have observed frames t_m) or noisy (i.e., the observed t_m for x_m is incorrect) during training, our approach can still extract useful signal.

3.2 Probabilistic Encoding

Our problem setup is that we have input text $\mathbf{x}$ describing some complex situation, paired with a partially observed sequence of frames $\mathbf{t}$. To account for when side knowledge is/isn't observed, we define a knowledge indicator set $\boldsymbol{l} = l_m|_{m=1}^{M}$ with $l_m \in \{0, 1\}$, where $l_m = 1$ denotes the external knowledge t_m is present and $l_m = 0$ means the external knowledge is not available (latent). To enable successful modeling, we introduce $\mathbf{z} = z_m|_{m=1}^{M}$, a set of M latent variables to first compress the information of the given inputs $\mathbf{x}$ and then be informative regarding $\mathbf{t}$. We define $p_\theta(\mathbf{z}|\mathbf{x})$, a probabilistic encoder from data points $\mathbf{x}$ to the latent variables $\mathbf{z}$, parameterized by a neural network θ.

We define a joint model over $\mathbf{t}, \mathbf{x}$ and $\mathbf{z}$ as $p(\mathbf{t}, \mathbf{x}, \mathbf{z}; \boldsymbol{l}) = p_{\mathcal{D}}(\mathbf{x}) \prod_m p_\theta(z_m|t_{m-1}, \mathbf{x}) p_\gamma\big(t_m|x_m, z_m; l_m\big)$, where $p_{\mathcal{D}}(\mathbf{x})$ is the empirical distribution over the input variables $\mathbf{x}$. Consistent with previous approaches, $p_\theta(z_m|t^{(s)}, \mathbf{x}; l_m)$ is a Gaussian. Similarly $p_\gamma\big(t_m|x_m, z_m; l_m\big)$ posits a distribution over semi-supervised latent knowledge $\mathbf{t}$ given $\mathbf{x}$ and latent variables $\mathbf{z}$. To learn richer representations, we force $\mathbf{z}$ and $\mathbf{t}$ to depend on one other in a sawtooth fashion: t_i depends on z_i, but z_i depends on t_{i-1}. Fig. 2 shows an illustration. This is a novel segmented, autoregressive sequencing for event modeling.

We define $r_\gamma(\mathbf{t}|\mathbf{z})$ as the *proposed distribution* that relates $\mathbf{z}$ to $\mathbf{t}$, where γ is computed as the outcome of a neural encoding of $\mathbf{z}$, NN($\mathbf{z}$). This distribution must be learned. For simplicity we omit the node index m when possible.

Revision Phase Incorporating the external knowledge in the training phase must satisfy two criteria: First, without observation ($l_m = 0$), we just rely on $r_\gamma(t_m|z_m)$. Second, when we have access to external knowledge, $p_{\mathcal{D}}(t_m|x_m)$ should be used for *guidance* and not discarding the proposed distribution. We define the *revised distribution* $p_\gamma\big(t_m|x_m, z_m; l_m\big)$ as a smoothed weighted average of the proposed distribution and the empirical distribution as $p_\gamma\big(t_m|x_m, z_m; l_m\big) = \frac{1}{1+\lambda l_m} r_\gamma(t_m|z_m) + \frac{\lambda l_m}{1+\lambda l_m} p_{\mathcal{D}}(t_m|x_m)$, where $\lambda \in \mathbb{R}^+$ is a weighting parameter for balancing the proposed and empirical distributions. In this setting, λ depends on the level of confidence in our external knowledge, where for less noisy knowledge, we can choose higher values for λ. In practice, we found that $\lambda = 1.0$ works reasonably.[1]

If side knowledge is absent ($l_m = 0$), we have $\hat{\lambda}_m = 1.0$ so $p_\gamma(t_m|x_m, z_m) = r_\gamma(t_m|z_m)$: the model just uses the proposed distribution. This allows gradients to propagate through the network. During the test phase we do not use any external knowledge, so the revised distribution $p_\gamma\big(\mathbf{t}|\mathbf{x}, \mathbf{z}; \boldsymbol{l}\big)$ reduces to the proposed distribution $r_\gamma(\mathbf{t}|\mathbf{z})$.

Analysis and Insights If we define $\hat{\lambda}_m = 1/(1 + \lambda l_m)$, we rewrite this as $p_\gamma\big(t_m|x_m, z_m; l_m\big) = \hat{\lambda}_m r_\gamma(t_m|z_m) + (1 - \hat{\lambda}_m) p_{\mathcal{D}}(t_m|x_m)$. This slight notation change is beneficial, as it lets us characterize the behavior of our revision step in terms of expected

[1] In dev experiments we tried various values of λ, both large and small. When λ was high (greater weight to the empirical distribution), the model over-emphasized this signal, which was detrimental at test time when we intentionally withheld this signal. In contrast, when λ is low (greater weight to the proposed distribution), there is less emphasis on the empirical distribution. However, as that empirical distribution could itself be quite sparse (due to ϵ), the model learns to disregard it. This highlights a strength in simplicity of the equal weighting.

observability of side knowledge. Specifically, in a semi-supervised setting, if the probability of observing any particular piece of side information can be modeled as $l \sim \text{Bern}(\epsilon)$, where ϵ is the observation probability, by marginalizing out l we have $\mathbb{E}_{l\sim\text{Bern}(\epsilon)} p_\gamma(t|x,z,l) = \hat{\epsilon} r_\gamma(t|z) + (1-\hat{\epsilon}) p_\mathcal{D}(t|x)$, where $\hat{\epsilon} = 1 - \left(\frac{\lambda}{1+\lambda}\right)\epsilon$. This indicates that with more observation, we rely more heavily on the empirical distribution and less on the proposed distribution. For space, see Appendix E for more details/the derivation.

Finally, our framework generalizes SSDVAE's parameter injection (see Appendix F):

Theorem 1. *If $r_\gamma(t|z) = Cat(\boldsymbol{\gamma})$, a categorical distribution with parameters γ, and the empirical distribution $p_\mathcal{D}(t|x)$ is a one-hot representation with $t_{k^*} = 1$, the revision step reduces to SSDVAE parameter injection.*

We have described how to guide the latent variable t in the encoding phase. Next we define our objective function to capture the background information and decoding $\mathbf{t}$ to effectively model event descriptions.

3.3 Training Objective and Decoding

After training, the model relies solely on the proposed distribution $r_\gamma(t_m|z_m)$ to predict t_m, implying the statistics of t_m will only depend on z_m. To capture the background information in $\mathbf{t}$, minimize $\mathcal{I}(\mathbf{x};\mathbf{t}|\mathbf{z})$ during training. In information theory terms, in the ideal situation where $\mathcal{I}(\mathbf{x};\mathbf{t}|\mathbf{z}) = 0$, there is not any residual information between $\mathbf{x}$ and $\mathbf{t}$ that is not captured by the latent representation $\mathbf{z}$ (Kirsch et al., 2020). Therefore without the need of $\mathbf{x}$ and $p_\mathcal{D}(\mathbf{t}|\mathbf{x})$, the latent $\mathbf{z}$ is enough to predict $\mathbf{t}$.

To be meaningfully learned, $\mathbf{t}$ must be informative enough to make some prediction or reconstruction. For clarity, we refer to the targets as $\mathbf{y}$, though for a task like language modeling, $\mathbf{x} = \mathbf{y}$. To achieve this learning, we maximize $\mathcal{I}(\mathbf{y},\mathbf{t})$.

Together, our ideal objective is $\mathcal{L} = -\mathcal{I}(\mathbf{y};\mathbf{t}) + \alpha\mathcal{I}(\mathbf{x};\mathbf{t}|\mathbf{z})$, where α is a tunable hyperparameter. This is difficult to optimize because within each $\mathcal{I}$ term there are intractable marginalizations (such as over $\mathbf{x}$). To understand why this is our ideal objective, we show that maximizing the intractable mutual information $\mathcal{I}(\mathbf{t};\mathbf{z})$ is inherently included in this unified objective for the reconstruction tasks:

Theorem 2. *For tasks where we maximize $\mathcal{I}(\mathbf{x};\mathbf{t})$: minimizing $\mathcal{I}(\mathbf{x};\mathbf{t}|\mathbf{z})$ leads to maximizing $\mathcal{I}(\mathbf{t};\mathbf{z})$.*

This theorem shows that reconstruction tasks like language modeling explicitly maximize the mutual information between different data representations, which is consistent with the InfoMax principle. See Appendix G for the proof. As a consequence of Theorem 2, we are maximizing the mutual information between two views of $\mathbf{x}$: compressed representation $\mathbf{z}$ and side information $\mathbf{t}$.

With that understanding, we proceed to a tractable approximation for our objective. Following Alemi et al. (2016), we have $-\mathcal{I}(\mathbf{y};\mathbf{t}) = -\mathbb{E}_{p(\mathbf{y},\mathbf{z},\mathbf{t})} \log \frac{p(\mathbf{y}|\mathbf{t})}{p(\mathbf{y})}$. We approximate this as

$$-\mathcal{I}(\mathbf{y};\mathbf{t}) \le -\mathbb{E}_{p(\mathbf{y},\mathbf{z},\mathbf{t})} \log \frac{q_\phi(\mathbf{y}|\mathbf{t})}{p(\mathbf{y})} \tag{2}$$

$$= \underbrace{-\mathbb{E}_{p(\mathbf{y},\mathbf{z},\mathbf{t})} \log q_\phi(\mathbf{y}|\mathbf{t})}_{\mathcal{L}_y} - H(\mathbf{y}). \tag{3}$$

Since $p(\mathbf{y}|\mathbf{t})$ is intractable, we approximate it via a decoder $q_\phi(\mathbf{y}|\mathbf{t})$ that can be computed by a neural network, denoted as ϕ. As the task entropy $H(\mathbf{y})$ is constant, we just minimize $\mathcal{L}_y$. While not explicitly reflected in Eq. 2, note that irrespective of whether side information is present or not, $\mathcal{I}(\mathbf{y};\mathbf{t})$ depends on $r_\gamma(\mathbf{t}|\mathbf{z})$. See Appendix B.1 for additional analysis of Eq. 3.

Updating Phase The term $\mathcal{I}(\mathbf{x};\mathbf{t}|\mathbf{z}) = \mathbb{E}_{p_\gamma(\mathbf{t},\mathbf{x},\mathbf{z})} \log \frac{p_\gamma(\mathbf{t}|\mathbf{x},\mathbf{z};\boldsymbol{l})}{p(\mathbf{t}|\mathbf{z})}$ makes $\mathbf{z}$ informative about $\mathbf{t}$. We approximate with a surrogate objective $\mathcal{L}_\mathcal{I}$:

$$\mathcal{I}(\mathbf{x};\mathbf{t}|\mathbf{z}) \le \mathbb{E}_{p(\mathbf{t},\mathbf{x},\mathbf{z})} \log \frac{p_\gamma(\mathbf{t}|\mathbf{x},\mathbf{z})}{r_\gamma(\mathbf{t}|\mathbf{z})} = \mathcal{L}_\mathcal{I}. \tag{4}$$

This surrogate objective encourages the proposed distribution $r_\gamma(\mathbf{t}|\mathbf{z})$ to be updated to be close to the revised distribution $p_\gamma(\mathbf{t}|\mathbf{x},\mathbf{z};\boldsymbol{l})$. After training, the proposed distribution $r_\gamma(\mathbf{t}|\mathbf{z})$ plays the role of $p_\gamma(\mathbf{t}|\mathbf{z},\mathbf{x})$ and does not need to explicitly use $p_\mathcal{D}(\mathbf{t}|\mathbf{x})$. Throughout, we refer to $\mathcal{L}_\mathcal{I}$ as *updating*. The updating objective for our model is given by

$$\mathcal{L}_\mathcal{I} = \sum_{m=1}^{M} \text{KL}\big(p_\gamma(t_m|x_m,z_m;l_m) \parallel r_\gamma(t_m|z_m)\big), \tag{5}$$

where by expanding further we obtain

$$\mathcal{L}_\mathcal{I} = \sum_{m=1}^{M} \Big[-H(t_m|x_m,z_m) + \hat{\lambda}_m H(t_m|z_m) - (1-\hat{\lambda}_m)\mathbb{E}_{p_\mathcal{D}(t_m|x_m)} \log r_\gamma(t_m|z_m) \Big]. \tag{6}$$

This sum is computed over observed nodes. For unsupervised nodes, the revised distribution and proposed distribution are equal and their KL terms are zero. The last term is a classification term.

Regularization To improve the model generalization ability, we introduce regularization terms for $\mathbf{z}$ and $\mathbf{t}$. We constrain the mutual information between data $\mathbf{x}$ and $\mathbf{z}$ latent representations, $\mathcal{I}(\mathbf{x};\mathbf{z}) = \mathbb{E}_{p(\mathbf{z},\mathbf{x})} \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})}$ as

$$\mathcal{I}(\mathbf{x};\mathbf{z}) \leq \underbrace{\mathrm{KL}\big(p_\theta(\mathbf{z}|\mathbf{x}) \parallel q(\mathbf{z})\big)}_{\mathcal{L}_z}, \tag{7}$$

where we introduce a variational distribution $q(\mathbf{z})$ due to intractability of $p(\mathbf{z})$. For simplicity we assume that $q(\mathbf{z})$ factorizes over independent Gaussian random variables as $q(\mathbf{z}) = \prod_{m=1}^{M} q(z_m)$, where the variational distribution over z_m is given by the unit Gaussian $z_m \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Here $\mathcal{L}_z$ is estimated with the standard Monte-Carlo sampling:

$$\mathcal{L}_z \approx \frac{1}{S} \sum_{m=1}^{M} \sum_{s=1}^{S} \mathrm{KL}\big(p_\theta(z_m|t_{m-1}^{(s)}, \mathbf{x}) \parallel q(z_m)\big). \tag{8}$$

We reduce the distance between the proposed distribution $r_\gamma(\mathbf{t}|\mathbf{z})$ and a uniform distribution $\mathcal{U}(\mathbf{t})$:

$$\mathcal{L}_t = \mathbb{E}_{p_\mathcal{D}(\mathbf{x})p_\theta(\mathbf{z}|\mathbf{x})r_\gamma(\mathbf{t}|\mathbf{z})} \log \frac{r_\gamma(\mathbf{t}|\mathbf{z})}{\mathcal{U}(\mathbf{t})}. \tag{9}$$

The Kullback-Leibler (KL) divergence terms in Eq. 7 and Eq. 9 help avoid overfitting. An alternative interpretation for these regularization terms is discarding the task-irrelevant information. We combine Eqs. 3, 4, 7 and 9 to at our final objective

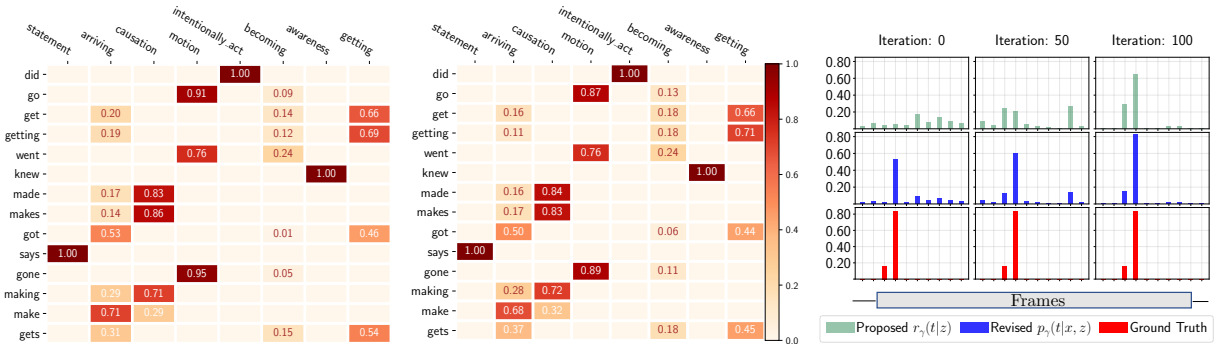
$$\mathcal{L}^U = \mathcal{L}_y + \alpha\mathcal{L}_\mathcal{I} + \beta\mathcal{L}_z + \zeta\mathcal{L}_t, \tag{10}$$

where β and ζ denote the trade-off parameters, and can be set empirically, as described in Appendix I.

3.4 Architecture For Event Modeling

To ensure fair comparisons, we focus on the reconstruction task similar to a β-VAE framework (Higgins et al., 2016). The overall structure is depicted in Fig. 2. Following previous work on event modeling (Pichotta and Mooney, 2016; Rezaee and Ferraro, 2021; Weber et al., 2018b; Gao et al., 2022), we represent each document $\mathbf{x}$ as a sequence of M events, where each event is a 4 tuple of predicate (verb), two main arguments (subject and object), and a modifier (if applicable). Each event is associated with a discrete semantic frame. E.g., *convicted man murdering of* is an event and the semantic frame is [VERDICT]. All the possible frames are collected in a vocabulary of size T. In this setting, we obtain a point estimate for $p_\mathcal{D}(t_m|x_m)$ as $\delta(x_m, t_m)$. Sampling from this empirical distribution outputs a one-hot vector of dimension T. The proposed distribution $r_\gamma(t|z)$ is a Gumbel-Softmax distribution. We found that the Gumbel-Softmax algorithm is particularly suitable for our task because it can effectively approximate discrete distributions and backpropagate gradients. While experiments with the Straight-Through Gumbel-Softmax (STGT) yielded near identical performance to the Gumbel-Softmax method, we opted for the latter. STGT generates one-hot vectors during the forward pass, but requires an approximation of the gradient using Gumbel-Softmax samples. By setting a low temperature of 0.5, the generated Gumbel-Softmax samples become almost identical to one-hot vectors, eliminating the need for gradient approximation. The effects on the gradient were carefully analyzed and the results are presented in Damadi and Shen (2022), where a comprehensive study of gradient properties was conducted. To learn richer representations, we define an embedding matrix $\mathbf{E} \in \mathbb{R}^{T \times d_t}$, to convert a simplex frame into a vector representation as $e_m = t_m^T \mathbf{E}$.

Encoding and Decoding With data point $\mathbf{x} \sim p_\mathcal{D}(\mathbf{x})$, we encode the whole sequence into recurrent hidden representations $\mathbf{H} = \{h\}_{t=1}^{T}$. For each event m, we draw Gaussian random variable $z_m \sim \mathcal{N}(\mu_m, \sigma_m)$ where μ_m and σ_m are the outputs of attention layers over frame embedding e_{m-1} and hidden states $\mathbf{H}$. We use a linear mapping over z_m to compute Gumbel softmax parameters for the proposed distribution $r_\gamma(t_m|z_m)$. Given the proposed distribution $r_\gamma(t_m|z_m)$ and the empirical distribution $p_\mathcal{D}(t_m|x_m)$, we first draw a Bernoulli sample $\hat{\lambda}_m$, then we draw knowledge sample from the mixture of probabilities by $t_m \sim p_\gamma(t_m|x_m, z_m; l_m)$. From the encoding phase, all the embedding vectors are gathered into $\mathbf{e} = \{e_m\}_{m=1}^{M}$. At generation time, analogously to the encoder, we use an attention layer over the decoder recurrent hidden state h and frame embeddings $\mathbf{e}$, resulting in decoder logits g. The generative distribution over possible next tokens is given by $x_t \sim p(x_t|g)$.

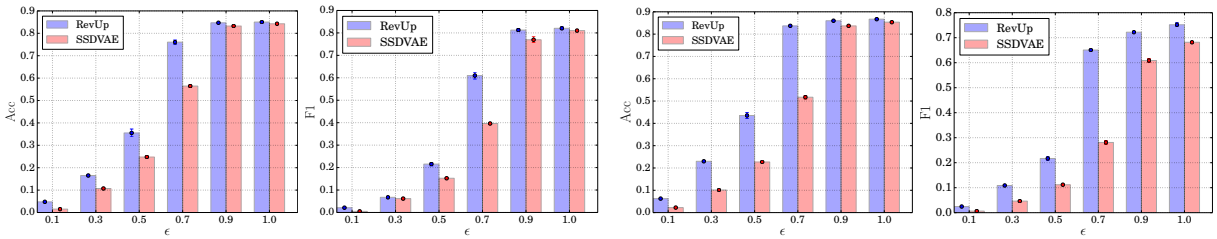


(a) Normalized samples from learned $r_\gamma(t|z)$ given predicates.

(b) Ground truth conditional distribution of semantic frames given predicates.

(c) The proposed and revised distributions converge to the ground truth distribution.

Figure 3: A demonstration of RevUp working on a small dataset with 10 frames. Ground truth frame distributions are shown in Fig. 3b, where in Fig. 3a we show that the distribution $r_\gamma(t|z)$ that RevUp learns very closely matches the ground truth. In Fig. 3c, we show how RevUp affects both the proposed and revised distributions. Initially, the $r_\gamma(t|z)$ prediction is random. After 50 iterations it is closer to the revised distribution $p_\gamma(t|z,x)$. After 100 iterations of training, both of them get close to the ground truth distribution.



(a) NYT Accuracy

(b) NYT F1 score

(c) Wikipedia Accuracy

(d) Wikipedia F1 score

Figure 4: RevUp vs SSDVAE accuracy for sequential semantic frame classifications for NYT and Wikipedia dataset. See Appendix H for full results. For all degree of observation probabilities, RevUp outperforms the baseline.

4 Experimental Results

Following previous work (Rezaee and Ferraro, 2021), we experiment on event modeling tasks using Concretely Annotated versions of New York Times Gigaword (NYT) and Wikipedia datasets (Ferraro et al., 2014). Both are English and have FrameNet semantic frames provided via SemaFor (Das et al., 2014). We perform a direct comparison with the previous state-of-the-art work (Rezaee and Ferraro, 2021) by using the same frame types, which were derived from the FrameNet annotations on Wikipedia articles. Training, validation and test splits for NYT and Wikipedia have 320k/17k/7k and 240k/8k/11k documents, respectively. For both the validation and test phases we set $l_m = 0$ (unsupervised). We average results over three runs (standard deviations in the appendix). We tuned the hyperparameters on the validation dataset. See Appendix I for additional implementation and data details.

4.1 Small Dataset Example

We first examine RevUp by examining its behavior on a small, focused example dataset. We sampled 400 newswire documents from the NYT dataset. We trained a RevUp model, with 10 semantic frame types (10 options for each t) and where each z was 100 dimensional. To obtain the ground-truth distributions of frames given events, we just focus on the predicates and collect all the semantic frames given that specific predicate. We carefully selected the frame types and data to reflect a diverse range of difficulties. For example, events including the *did* predicate are always associated with the INTENTIONALLY_ACT frame and the corresponding semantic frames for the *gets* predicate are ARRIVING, BECOMING and GETTING. RevUp predictions are acquired from the revised distribution $r_\gamma(t|z)$: given an event x we first draw a Gaussian sample z and then use $\text{argmax}(t)$ to find the proposed index. Finally, we normalize across all the proposed frames. We visualize the revision and update steps for the predicate "*made*" in Fig. 3c.

As evidenced by Eq. 4, the minimization of the KL-divergence between the proposed distribution $r_\gamma(t|z)$ and the revised distribution $p_\gamma(t|x,z)$ results in the elimination of non-essential information. Fig. 3c gives a visual representation of this

phenomenon. At iteration 0, the proposed distribution $p_\gamma(t|x,z)$ is almost uniform, while the revised distribution $r_\gamma(t|z)$ is more aligned with the ground distribution. With additional iterations, reaching 100, the gap between these distributions reduces, ultimately leading to convergence with the ground truth for both distributions.

Additionally, our framework is able to effectively capture the conditional distribution of frames given predicates in diverse scenarios, as demonstrated in the heatmap figures in Figs. 3a and 3b. For instance, certain predicates such as "did," "says," and "knew" have a single associated frame of "intentionally_act," "awareness," and "statement," respectively. Meanwhile, the predicate "made" has two possible semantic frames of "arriving" and "causation." In some cases, such as the predicate "get," there are even three possible frames of "arriving," "becoming," and "getting." The comparison between the ground truth distribution of frames given predicates and the normalized samples highlights the accuracy of our model in capturing these conditional distributions, with minimal error.

4.2 Baselines

We compare the proposed RevUp method with the following event modeling approaches. (a) **RNNLM** (Pichotta and Mooney, 2016): A Bidirectional GRU cell with two layers, hidden dimension of 512, gradient clipping at 5 and Glove 300 embeddings to represent words. We used the implementation provided in (Weber et al., 2018b). (b) **HAQAE** (Weber et al., 2018b): An unsupervised VAE-based method for script learning and generation. They use vector quantization (VQ–VAE) to define a hierarchical discrete latent space. (c) **SSDVAE** (Rezaee and Ferraro, 2021) A semi–supervised VAE-based model. They utilize Gumbel softmax and use the parameter injection to incorporate the side knowledge. The architecture of our model largely follows the SSDVAE and HAQAE models with minor modification; see Appendix I.

4.3 Effect of Noisy Knowledge

We empirically compare the predictive performance of RevUp and SSDVAE with noisy knowledge. To do so, in a fully supervised setting, for each event x_m, instead of using the associated semantic frame t_m, with probability η we replace t_m with a random semantic frame $\hat{t}_m$. We train both models with this new training dataset. During the testing phase, we compare the predicted knowledge with ground-truth one. Results in Table 1 show that classification and parameter injection in SSDVAE are not enough for capturing knowledge. These results validate the effectiveness of our information capturing strategy when the external knowledge is noisy. As we increase η, SSDVAE performance degrades much more than RevUp. As an instance when $\eta = 0.9$, SSDVAE's accuracy is almost zero but our proposed model can achieve 0.41.

model	η	Wiki		NYT	
		Acc	F1	Acc	F1
SSDVAE	0.1	0.77	0.43	0.76	0.55
RevUp		**0.85**	**0.58**	**0.83**	**0.71**
SSDVAE	0.2	0.69	0.35	0.67	0.44
RevUp		**0.82**	**0.52**	**0.80**	**0.63**
SSDVAE	0.3	0.60	0.28	0.58	0.36
RevUp		**0.79**	**0.45**	**0.77**	**0.58**
SSDVAE	0.5	0.41	0.17	0.41	0.23
RevUp		**0.72**	**0.36**	**0.71**	**0.48**
SSDVAE	0.7	0.23	0.09	0.22	0.11
RevUp		**0.64**	**0.29**	**0.63**	**0.37**
SSDVAE	0.9	0.02	0.00	0.02	0.00
RevUp		**0.41**	**0.09**	**0.25**	**0.06**

Table 1: Effect of Noise on robustness. We present standard deviations in Appendix H.

model	ϵ	Wiki	NYT
RNNLM	-	56.96	64.57
HAQAE	-	39.47	50.10
SSDVAE	0.0	39.75	47.50
RevUp		**39.48**	**45.36**
SSDVAE	0.1	39.73	45.91
RevUp		**33.34**	**44.87**
SSDVAE	0.7	36.79	44.79
RevUp		**33.20**	**41.74**
SSDVAE	1.0	30.69	36.96
RevUp		**28.33**	**34.85**

Table 2: Test perplexity results, varying the percent ϵ of side knowledge that was observed during training. We present standard deviations in Appendix H (table 4).

4.4 Effect of Incomplete Knowledge

We consider the case of semi-supervised learning, where with probability ϵ a node t_m is observed. We report the classifications on the latent nodes as $\mathrm{KL}\big(p_\mathcal{D}(\mathbf{t}|\mathbf{x}) \parallel r_\gamma(\mathbf{t}|\mathbf{x})\big) \approx -\mathbb{E}_{p_\mathcal{D}(\mathbf{x})p_\theta(\mathbf{z}|\mathbf{x})p_\mathcal{D}(\mathbf{t}|\mathbf{x})} \log r_\gamma(\mathbf{t}|\mathbf{z})$. The results are shown in Fig. 4. We report two widely-used classification metrics including accuracy and F1 to evaluate the performance of all methods. SSDVAE just relies on the guidance and classification but RevUp also uses the updating phase to shift the available knowledge from side information into latent variables. Thus the results demonstrate the superiorities of RevUp to predict knowledge when

they are partially observed, attributable to its novel information injection and learning.

To investigate whether the proposed approach works for task representation, we compare the perplexity of our approach to prior work in Table 2. Perplexity has been commonly used in the literature, which allows us to provide a fair comparison with previous efforts. We investigate the effect of supervision with various observation probabilities $\epsilon = 0.0$ (unsupervised), 0.1, 0.7 and 1.0 (fully supervised) on the generated samples from the model. We see that our model is able to obtain lower perplexity scores than the previous event modeling methods. We observe as ϵ increases, the performance of our proposed model improves. For each observation probability, our method outperforms SSDVAE. The results demonstrate that our method achieves state-of-the-art performance with large margins from the baselines.

5 Related Work

Information Bottleneck The concept of using side information for discrete source coding was explored by Wyner (1975); Wyner and Ziv (1976). The Information Bottleneck (IB) principle was then introduced by (Tishby et al., 2000) to compress input variables while predicting a target. Chechik and Tishby (2002) proposed incorporating negative side information through an auxiliary loss in a supervised manner. Our method stands apart by handling both supervised and semi-supervised settings with ease. The Variational Information Bottleneck (VIB) was introduced by Alemi et al. (2016), which improved the IB estimation through amortized variational methods. Voloshynovskiy et al. (2020) extended the VIB method for semi-supervised classification. In relation to our work, there have been numerous studies focused on maximizing the mutual information between different views and discarding the non-shared information (Federici et al., 2019; Wan et al., 2021; Wang et al., 2019; Mao et al., 2021; Yan et al., 2019).

Variational Autoencoders (VAEs) Kingma and Welling (2013) and Rezende et al. (2014) introduced the reparameterization technique for variational inference. Recently, Huang and Xiao (2021) applied adversarial training to enhance the ability of VAE in processing sequential data, while Qiu et al. (2020) explored the possibility of VAE handling multi-view temporal data. Our focus is limited to objectives that can be expressed with discrete latent variables.

Jang et al. (2016) and Maddison et al. (2016) independently presented the Gumbel-Softmax distribution, which allows backpropagation through latent discrete variables in variational autoencoders. For more information, refer to the review by Huijben et al. (2022). Van Den Oord et al. (2017) proposed VQ-VAE, which uses vector quantization to represent discrete values with embeddings.

Kingma et al. (2014) proposed a VAE for semi-supervised classification, where labels in unsupervised settings are treated as latent variables and predicted when available. Joy et al. (2022) later improved this approach by separating latent variables to differentiate label characteristics from values. Lo and Lim (2020) modified the VAE objective to incorporate external knowledge with embedding vectors, and Feng et al. (2020) proposed a new ELBO to incorporate label prediction loss.

Event Modeling In recent years, much research has been dedicated to the challenge of modeling the sequence of events (Weber et al., 2018b; Rezaee and Ferraro, 2021; Gao et al., 2022). One such contribution was made by Weber et al. (2018a), who introduced a tensor-based composition to effectively capture semantic event relations. Gao et al. (2022) proposed a self-supervised contrastive learning approach based on co-occurring events. Another approach suggested by Weber et al. (2018b) involves the use of Recurrent Neural Networks (RNNs) to model the hierarchical latent structures that exist in sequences of events. This methodology was further developed in a subsequent study by Rezaee and Ferraro (2021), where external knowledge was incorporated into the latent layer for enhanced modeling, which is highly relevant to our proposed approach.

6 Conclusion

We show how to incorporate noisy partially observed side knowledge source along with latent variables. To do so, we generalized the main idea of parameter injection and maximizing the mutual information between external knowledge and latent variables. Our experiments show that our approach is more robust to noisy knowledge and outperform other baselines for the event modeling task

Acknowledgments

We would like to thank the anonymous reviewers for their comments, questions, and suggestions.

This material is based in part upon work supported by the National Science Foundation under Grant No. IIS-2024878. Some experiments were conducted on the UMBC HPCF, supported by the National Science Foundation under Grant No. CNS-1920079. This material is also based on research that is in part supported by the Army Research Laboratory, Grant No. W911NF2120076, and by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of the Air Force Research Laboratory (AFRL), DARPA, or the U.S. Government.

7 Limitations

Some limitations of our work are that multiple hyper parameters should be tuned for each task. While we provide guidance and insights into effective settings (and some intuition as to why), we acknowledge that the settings may be domain dependent.

Because we use semantic frames as our side knowledge, our focus is on improving the representation and use of discrete latent variables. While current NLP approaches have often focused on text-to-text methods for input and output, and individual words in text can be considered a form of discrete latent variables, we note that these methods are driven by large continuous embedding methods. While we believe this work can be extended to continuous cases, the approximations did make use of aspects of discrete variables, and they would need to be re-derived.

RevUp depends on mixing in statistics and learned representations from external side knowledge. While we envision this side knowledge as containing useful generalizations and semantic information, such resources could encode overly broad generalizations or other biases. While the degree of this mixture can be adjusted, imperfections or biases in the external knowledge could be captured and propagated through RevUp.

We focus on the task of event modeling, but we believe RevUp represents a step towards improving settings where noisy side information is available.

References

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2016. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

Sarkhan Badirli, Zeynep Akata, George Mohler, Christine Picard, and Mehmet M Dundar. 2021. Fine-grained zero-shot learning with dna as side information. *Advances in Neural Information Processing Systems*, 34:19352–19362.

Gal Chechik and Naftali Tishby. 2002. Extracting relevant structures with side information. *Advances in Neural Information Processing Systems*, 15.

Mingda Chen, Qingming Tang, Karen Livescu, and Kevin Gimpel. 2019. Variational sequential labelers for semi-supervised learning. *arXiv preprint arXiv:1906.09535*.

Muhao Chen, Hongming Zhang, Haoyu Wang, and Dan Roth. 2020. What are you trying to do? semantic typing of event processes. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 531–542.

Thomas M Cover. 1999. *Elements of information theory*. John Wiley & Sons.

Saeed Damadi and Jinglai Shen. 2022. Gradient properties of hard thresholding operator. *arXiv preprint arXiv:2209.08247*.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Marco Federici, Anjan Dutta, Patrick Forré, Nate Kushman, and Zeynep Akata. 2019. Learning robust representations via multi-view information bottleneck. In *International Conference on Learning Representations*.

Hao-Zhe Feng, Kezhi Kong, Minghao Chen, Tianye Zhang, Minfeng Zhu, and Wei Chen. 2020. Shot-vae: semi-supervised deep generative models with label-aware elbo approximations. *CoRR abs/2011.10684. URL https://arxiv. org/abs/2011*, 10684.

Francis Ferraro, Max Thomas, Matthew R Gormley, Travis Wolfe, Craig Harman, and Benjamin Van Durme. 2014. Concretely annotated corpora. In *4th Workshop on Automated Knowledge Base Construction (AKBC)*.

Francis Ferraro and Benjamin Van Durme. 2016. A Unified Bayesian Model of Scripts, Frames and Language. In *Proceedings of the 30th Conference on Artificial Intelligence (AAAI)*, pages 2601–2607, Phoenix, Arizona. Association for the Advancement of Artificial Intelligence.

Arthur Flajolet and Patrick Jaillet. 2017. Real-time bidding with side information. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jun Gao, Wei Wang, Changlong Yu, Huan Zhao, Wilfred Ng, and Ruifeng Xu. 2022. Improving event representation via simultaneous weakly supervised contrastive learning and clustering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3036–3049, Dublin, Ireland. Association for Computational Linguistics.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework.

Jin Huang and Ming Xiao. 2021. Regularized sequential latent variable models with adversarial neural networks. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 834–839. IEEE.

Iris AM Huijben, Wouter Kool, Max Benedikt Paulus, and Ruud JG Van Sloun. 2022. A review of the gumbel-max trick and its extensions for discrete stochasticity in machine learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Tom Joy, Yuge Shi, Philip Torr, Tom Rainforth, Sebastian M Schmon, and Siddharth N. 2022. Learning multimodal VAEs through mutual supervision. In *International Conference on Learning Representations*.

Alex Judea and Michael Strube. 2015. Event extraction as frame-semantic parsing. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 159–164, Denver, Colorado. Association for Computational Linguistics.

Di Kang, Debarun Dhar, and Antoni Chan. 2017. Incorporating side information by adaptive convolution. *Advances in Neural Information Processing Systems*, 30.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27.

Andreas Kirsch, Clare Lyle, and Yarin Gal. 2020. Unpacking information bottlenecks: Surrogate objectives for deep learning.

Lingpeng Kong, Cyprien de Masson d'Autume, Lei Yu, Wang Ling, Zihang Dai, and Dani Yogatama. 2019. A mutual information maximization perspective of language representation learning. In *International Conference on Learning Representations*.

R. Linsker. 1988. Self-organization in a perceptual network. *Computer*, 21(3):105–117.

Pei-Chi Lo and Ee-Peng Lim. 2020. Co-embedding attributed networks with external knowledge.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Yiqiao Mao, Xiaoqiang Yan, Qiang Guo, and Yangdong Ye. 2021. Deep mutual information maximin for cross-modal clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8893–8901.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Ankur Padia, Francis Ferraro, and Tim Finin. 2018. SURFACE: Semantically rich fact validation with explanations. ArXiv:1810.13223.

Karl Pichotta and Raymond Mooney. 2016. Learning statistical scripts with lstm recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Lin Qiu, Vernon M Chinchilli, and Lin Lin. 2020. Interpretable deep representation learning from temporal multi-view data. *arXiv preprint arXiv:2005.05210*.

Mehdi Rezaee and Francis Ferraro. 2021. Event representation with sequential, semi-supervised discrete variables. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4701–4716.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR.

Naftali Tishby, Fernando C. Pereira, and William Bialek. 1999. The information bottleneck method. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377.

Naftali Tishby, Fernando C. Pereira, and William Bialek. 2000. The information bottleneck method.

Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Slava Voloshynovskiy, Olga Taran, Mouad Kondah, Taras Holotyak, and Danilo Rezende. 2020. Variational information bottleneck for semi-supervised classification. *Entropy*, 22(9):943.

Zhibin Wan, Changqing Zhang, Pengfei Zhu, and Qinghua Hu. 2021. Multi-view information-bottleneck representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10085–10092.

Qi Wang, Claire Boudreau, Qixing Luo, Pang-Ning Tan, and Jiayu Zhou. 2019. Deep multi-view information bottleneck. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 37–45. SIAM.

Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. 2018a. Event representations with tensor-based compositions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Noah Weber, Leena Shekhar, Niranjan Balasubramanian, and Nathanael Chambers. 2018b. Hierarchical quantized representations for script generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3783–3792.

A. Wyner. 1975. On source coding with side information at the decoder. *IEEE Transactions on Information Theory*, 21(3):294–300.

A. Wyner and J. Ziv. 1976. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on Information Theory*, 22(1):1–10.

Patrick Xia, Guanghui Qin, Siddharth Vashishtha, Yunmo Chen, Tongfei Chen, Chandler May, Craig Harman, Kyle Rawlins, Aaron Steven White, and Benjamin Van Durme. 2021. LOME: Large ontology multilingual extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*.

Xiaoqiang Yan, Yangdong Ye, Yiqiao Mao, and Hui Yu. 2019. Shared-private information bottleneck method for cross-modal clustering. *IEEE Access*, 7:36045–36056.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Qi Zhang, Wei Lin, and Antoni B Chan. 2021. Cross-view cross-scene multi-view crowd counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 557–567.

A Information Bottleneck Principle (IB)

Given its importance to our effort, we restate the information bottleneck principle, and discuss how it ties in with our loss formulation. Then, in subsequent appendices (App. B to D) we step through individual subloss terms.

Given the original input data $\mathbf{x}$ with the target $\mathbf{y}$, IB aims to extract a compact representation of $\mathbf{x}$ that is most informative about the target $\mathbf{y}$ via maximizing $\mathcal{I}(y;z) - \beta\mathcal{I}(x;z)$. The first term $\mathcal{I}(y;z)$ motivates the model to predict y, whilst the second term $\mathcal{I}(x;z)$ aims at discarding irrelevant information from the input x. The hyperparameter β can be set or tuned to adjust the relative importance of discarding irrelevant information. In our framework, we have a generalized version of IB as follows

$$\begin{aligned}\mathcal{L} = &- \mathcal{I}(\mathbf{y};\mathbf{t}) + \alpha\mathcal{I}(\mathbf{x};\mathbf{t}|\mathbf{z}) \\ &+ \beta\mathcal{I}(\mathbf{x};\mathbf{z}) + \zeta\mathrm{KL}\big(r_\gamma(\mathbf{t}|\mathbf{z}) \parallel \mathcal{U}(\mathbf{t})\big). \quad (11)\end{aligned}$$

Our generalized version accounts for additional random variables, and the associated dependencies. Here, $r_\gamma(\mathbf{t}|\mathbf{z})$ is the proposed distribution, $\mathcal{U}$ is a uniform distribution over $\mathbf{t}$. The KL term helps avoid overfitting.

B Regularization and Task Representation Derivation

Throughout we use the following inequality

$$\mathbb{E}_{p(\mathbf{z},\mathbf{x})} \log \frac{q(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} \leq \mathbb{E}_{p_{\mathcal{D}(\mathbf{x})}} \log \underbrace{\mathbb{E}_{p_\theta}(\mathbf{z}|\mathbf{x}) \frac{q(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}}_{1} = 0. \quad (12)$$

The regularization term $\mathcal{I}(\mathbf{x};\mathbf{z})$ is

$$\begin{aligned}\mathcal{I}(\mathbf{x};\mathbf{z}) &= \mathbb{E}_{p(\mathbf{z},\mathbf{x})} \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} \\ &\leq \mathbb{E}_{p(\mathbf{z},\mathbf{x})} \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} \quad (13)\end{aligned}$$

where since $p(\mathbf{z}) = \int p_\theta(\mathbf{z}|\mathbf{x}) p_{\mathcal{D}}(\mathbf{x})\, dx$ is intractable we use the inequality in Eq. 12. Now, we can approximate Eq. 13 as follows

$$\begin{aligned}&\mathbb{E}_{p_{\mathcal{D}}(\mathbf{x}) p_\theta(\mathbf{z}|\mathbf{x})} \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} \quad (14)\\ &= \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x}) p_\theta(\mathbf{z}|\mathbf{x})} \log \frac{\prod_{m=1}^M p_\theta(z_m|t_{m-1},\mathbf{x})}{\prod_{m=1}^M q(z_m)} \\ &= \sum_{m=1}^M \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x}) p_\theta(\mathbf{z}|\mathbf{x})} \log \frac{p_\theta(z_m|t_{m-1},\mathbf{x})}{q(z_m)} \\ &\approx \frac{1}{S} \sum_{m=1}^M \sum_{s=1}^S \mathrm{KL}\big(p_\theta(z_m|t_{m-1}^{(s)},\mathbf{x}) \parallel q(z_m)\big), \quad (15)\end{aligned}$$

where for simplicity we assume that $q(\mathbf{z})$ factorizes over independent random variables as $q(\mathbf{z}) = q(z_1, z_2, \ldots, z_M) = \prod_{m=1}^M q(z_m)$. We parameterize each z_m via a multivariate Normal, $q(z_m) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $p_\theta(z_m|z_{m-1}^{(s)}, \mathbf{x}) = \mathcal{N}(\mu_m, \Sigma_m)$:

$$\begin{aligned}\mu_m &= f^\mu(z_{m-1}^{(s)}, \mathbf{x}) \\ \Sigma_m &= f^\Sigma(z_{m-1}^{(s)}, \mathbf{x}),\end{aligned}$$

where both f^μ and f^Σ are neural networks. With this formulation, we can calculate the KL-divergence terms of Eq. 15 in closed forms.

B.1 Approximating $\mathcal{I}(\mathbf{y};\mathbf{t})$ (Eq. 3)

In the same way for the task representation, we have

$$\begin{aligned}-\mathcal{I}(\mathbf{y};\mathbf{t}) &= -\mathbb{E}_{p(\mathbf{y},\mathbf{z},\mathbf{t})} \log \frac{p(\mathbf{y}|\mathbf{t})}{p(\mathbf{y})} \\ &\leq -\mathbb{E}_{p(\mathbf{y},\mathbf{z},\mathbf{t})} \log \frac{q_\phi(\mathbf{y}|\mathbf{t})}{p(\mathbf{y})} \\ &\leq -\mathbb{E}_{p(\mathbf{y},\mathbf{z},\mathbf{t})} \log q_\phi(\mathbf{y}|\mathbf{t}) + H(\mathbf{y}), \quad (16)\end{aligned}$$

where $H(\mathbf{y})$ is constant and we ignore it. We have:

$$\begin{aligned}&\mathbb{E}_{p(\mathbf{y},\mathbf{z},\mathbf{t})} \log q_\phi(\mathbf{y}|\mathbf{t}) \\ &= \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x},\mathbf{y}) p_\theta(\mathbf{z}|\mathbf{x}) p_\gamma(\mathbf{t}|\mathbf{z})} \log q_\phi(\mathbf{y}|\mathbf{t}), \quad (17)\\ &\approx \frac{1}{S_z}\frac{1}{S_t} \sum_{s_z=1}^{S_z} \sum_{s_t=1}^{S_t} \log q_\phi(\mathbf{y}|\mathbf{t}^{(s_t)}), \quad (18)\end{aligned}$$

where we first sample $\mathbf{x}, \mathbf{y} \sim p_{\mathcal{D}}(\mathbf{x}, \mathbf{y})$, then $\mathbf{z}^{(s_z)} \sim p_\theta(\mathbf{z}|\mathbf{x})$ and finally $\mathbf{t}^{(s_t)} \sim r_\gamma(\mathbf{t}|\mathbf{z}^{(s_z)})$. In this approximation S_z and S_t are the total number of samples for $\mathbf{z}$ and $\mathbf{t}$ respectively.

We wish to clarify why this term is included in the objective and how optimizing it affects the

model parameters. The answer is that in computing $\mathcal{I}(\mathbf{y};\mathbf{t})$, the distribution used to compute the mutual information's expectation depends on the proposed distribution r_γ. This is by construction and irrespective of whether side information is present or not. This is where the model parameters come in.

Specifically, while by definition $\mathcal{I}(\mathbf{y};\mathbf{t}) = \mathbb{E}_{\mathbf{p(y,t)}} \frac{\mathbf{log p(y|t)}}{\mathbf{p(y)}}$, because $p(|\mathbf{t})$ does not depend on a (free) $\mathbf{z}$, we can compute $mut(\mathbf{y};\mathbf{t}) = \mathbb{E}_{\mathbf{p(y,z,t)}} \frac{\mathbf{log p(y|t)}}{\mathbf{p(y)}}$. This follows from the fact that for random variables u and v, $\mathbb{E}_{p(u,v)}[f(u)] = \mathbb{E}_{p(u)}[f(u)]$. With this, we can then write

$$\mathcal{I}(\mathbf{y};\mathbf{t}) = \mathbb{E}_{p(\mathbf{y},\mathbf{t})} \log \frac{p(\mathbf{y}|\mathbf{t})}{p(\mathbf{y})}$$

$$= \mathbb{E}_{p(\mathbf{y},\mathbf{t},\mathbf{x},\mathbf{z})} \log \frac{p(\mathbf{y}|\mathbf{t})}{p(\mathbf{y})} \tag{19}$$

$$= \mathbb{E}_{\left\{p_{\mathcal{D}}(\mathbf{x},\mathbf{y})p(\mathbf{z},\mathbf{t}|\mathbf{x})\right\}} \log \frac{p(\mathbf{y}|\mathbf{t})}{p(\mathbf{y})} \tag{20}$$

$$= \mathbb{E}_{\left\{p_{\mathcal{D}}(\mathbf{x},\mathbf{y}) \prod_{m=1}^{M} p_\theta(z_m|t_{m-1},\mathbf{x}) p_\gamma(t_m|x_m,z_m;l_m)\right\}} \tag{21}$$

$$\log \frac{p(\mathbf{y}|\mathbf{t})}{p(\mathbf{y})}. \tag{22}$$

This red term, $p_\gamma(t_m|x_m, z_m; l_m)$, is defined as the interpolation of a proposal distribution $r_\gamma(t_m|z_m)$ and an empirical distribution $t_m|x_m$. Defined as a GumbelSoftmax, this proposal distribution explicitly allows uncertainty. Therefore, even when side information is present, we sample the value of t_m to approximate that final expectation.

Finally, as mentioned in the paper, this last line is intractable, so we learn an approximation $q_\phi(y|t)$, which introduces additional model parameters to learn. We compute q_ϕ using the sampled values $t_1 ... t_M$.

C Update Step Upperbound

In this section, we show how we approximate the update term $\mathcal{I}(\mathbf{x};\mathbf{z}|\mathbf{z})$ in Eq. 4. We have the following

$$\mathcal{I}(\mathbf{t};\mathbf{x}|\mathbf{z}) = \mathbb{E}_{p_\gamma(\mathbf{t},\mathbf{z},\mathbf{x})} \log \frac{p_\gamma(\mathbf{t},\mathbf{x}|\mathbf{z})}{p(\mathbf{t}|\mathbf{z})p(\mathbf{x}|\mathbf{z})}$$

$$= \mathbb{E}_{p_\gamma(\mathbf{t},\mathbf{x},\mathbf{z})} \log \frac{p_\gamma(\mathbf{t}|\mathbf{x},\mathbf{z})p(\mathbf{x}|\mathbf{z})}{p(\mathbf{t}|\mathbf{z})p(\mathbf{x}|\mathbf{z})}$$

$$= \mathbb{E}_{p_\gamma(\mathbf{t},\mathbf{x},\mathbf{z})} \log \frac{p_\gamma(\mathbf{t}|\mathbf{x},\mathbf{z})}{p(\mathbf{t}|\mathbf{z})} \tag{23}$$

$$\leq \mathbb{E}_{p_\gamma(\mathbf{t},\mathbf{x},\mathbf{z})} \log \frac{p_\gamma(\mathbf{t}|\mathbf{x},\mathbf{z})}{r_\gamma(\mathbf{t}|\mathbf{z})}, \tag{24}$$

First, we should note that in Eq. 23, the distribution $p(\mathbf{t}|\mathbf{z})$ is not equal to $r_\gamma(\mathbf{t}|\mathbf{z})$, because for each node m, we have

$$\begin{aligned} p(t_m|z_m) &= \int p(t_m, x_m|z_m)\, dx_m \\ &= \int p(t_m|x_m, z_m)p(x_m|z_m)\, dx_m \\ &= \int \hat{\lambda}_m r_\gamma(t_m|z_m)p(x_m|z_m)\, dx_m + \\ &\int (1-\hat{\lambda}_m)p_{\mathcal{D}}(t_m|x_m)p(x_m|z_m)\, dx_m \\ &\neq r_\gamma(t_m|z_m). \end{aligned} \tag{25}$$

Our the last step of estimation in Eq. 24 is

$$\mathbb{E}_{p_\gamma(\mathbf{t},\mathbf{x},\mathbf{z})} \log \frac{p_\gamma(\mathbf{t}|\mathbf{x},\mathbf{z})}{p(\mathbf{t}|\mathbf{z})} \leq \mathbb{E}_{p_\gamma(\mathbf{t},\mathbf{x},\mathbf{z})} \log \frac{p_\gamma(\mathbf{t}|\mathbf{x},\mathbf{z})}{r_\gamma(\mathbf{t}|\mathbf{z})}. \tag{26}$$

The proof for Eq. 26 is as follows.

$$\begin{aligned} &\mathbb{E}_{p(\mathbf{t},\mathbf{x},\mathbf{z})} \log \frac{r_\gamma(\mathbf{t}|\mathbf{z})}{p(\mathbf{t}|\mathbf{z})} \\ &= \mathbb{E}_{p(\mathbf{t},\mathbf{z})} \log \frac{r_\gamma(\mathbf{t}|\mathbf{z})}{p(\mathbf{t}|\mathbf{z})} \\ &= \mathbb{E}_{p(\mathbf{z})}\mathbb{E}_{p(\mathbf{t}|\mathbf{z})} \log \frac{r_\gamma(\mathbf{t}|\mathbf{z})}{p(\mathbf{t}|\mathbf{z})} \\ &\leq \mathbb{E}_{p(\mathbf{z})} \log \Big\{ \underbrace{\mathbb{E}_{p(\mathbf{t}|\mathbf{z})} \frac{r_\gamma(\mathbf{t}|\mathbf{z})}{p(\mathbf{t}|\mathbf{z})}}_{1} \Big\} \\ &= 0 \end{aligned}$$

D Update State Approximation for Discrete Case

We show how we approximate the upperbound presented in Eq. 24.

$$\begin{aligned} &\mathbb{E}_{p_\gamma(\mathbf{t},\mathbf{z},\mathbf{x})} \log \frac{p_\gamma(\mathbf{t}|\mathbf{z},\mathbf{x})}{r_\gamma(\mathbf{t}|\mathbf{z})} \\ &= \underbrace{\mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})p_\theta(\mathbf{z}|\mathbf{x})p_\gamma(\mathbf{t}|\mathbf{z},\mathbf{x})} \log p_\gamma(\mathbf{t}|\mathbf{z},\mathbf{x})}_{\text{A}_1} \\ &- \underbrace{\mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})p_\theta(\mathbf{z}|\mathbf{x})p_\gamma(\mathbf{t}|\mathbf{z},\mathbf{x})} \log r_\gamma(\mathbf{t}|\mathbf{z})}_{\text{A}_2}. \end{aligned} \tag{27}$$

Here $\textcircled{A_1}$ is the negative entropy of the revised distribution $p_\gamma(\mathbf{t}|\mathbf{x},\mathbf{z})$:

$$\textcircled{A_1} = -H(\mathbf{t}|\mathbf{x},\mathbf{z}) \tag{28}$$

$$= -\sum_{m=1} H(t_m|x_m,z_m). \tag{29}$$

For the second term we have:

$$\textcircled{A_2} = \mathbb{E}_{p_\mathcal{D}(\mathbf{x})}\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}\Big[\hat{\lambda}\mathbb{E}_{r_\gamma(\mathbf{t}|\mathbf{z})} + (1-\hat{\lambda})\mathbb{E}_{p_\mathcal{D}(\mathbf{t}|\mathbf{x})}\Big]\log r_\gamma(\mathbf{t}|\mathbf{z})$$

$$= \hat{\lambda}\underbrace{\mathbb{E}_{p_\mathcal{D}(\mathbf{x})}\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}\mathbb{E}_{r_\gamma(\mathbf{t}|\mathbf{z})}\log r_\gamma(\mathbf{t}|\mathbf{z})}_{\textcircled{A_{21}}}$$

$$+ (1-\hat{\lambda})\underbrace{\mathbb{E}_{p_\mathcal{D}(\mathbf{x})}\mathbb{E}_{p_\mathcal{D}(\mathbf{t}|\mathbf{x})}\log r_\gamma(\mathbf{t}|\mathbf{z})}_{\textcircled{A_{22}}}. \tag{30}$$

Similarly $\textcircled{A_{21}}$ is the entropy of the proposed distribution $r_\gamma(\mathbf{t}|\mathbf{z})$:

$$\textcircled{A_{21}} = -H(\mathbf{t}|\mathbf{z}) \tag{31}$$

$$= -\sum_{m=1}^{M} H(t_m|z_m) \tag{32}$$

Also $\textcircled{B_{22}}$ guides the model to discriminator $\mathbf{t}$ given $\mathbf{z}$ which is a classifier. To compute the entropy terms, in the discrete case (Gumbel Softmax), we use Monte Carlo sampling as follows

$$H(t_m|x_m,z_m) \approx \frac{1}{S}\sum_{n=1}^{N}\sum_{s=1}^{S}\hat{\boldsymbol{\gamma}}_m^{\top(s)}\log\hat{\boldsymbol{\gamma}}_m^{(s)}, \tag{33}$$

$$H(t_m|z_m) \approx \frac{1}{S}\sum_{s=1}^{S}\boldsymbol{\gamma}_m^{\top(s)}\log\boldsymbol{\gamma}_m^{(s)}, \tag{34}$$

where S is the number of point estimates and $\hat{\gamma}_m$ is derived from Eq. 35. Here, we have defined $\hat{\boldsymbol{\gamma}}$ to be $[\hat{\lambda}\gamma_1, \hat{\lambda}\gamma_2, \ldots, \hat{\lambda}\gamma_{k^*} + (1-\hat{\lambda}), \ldots, \hat{\lambda}\gamma_K]$. The $\hat{\boldsymbol{\gamma}}$ can be thought of as a mixture of the $\boldsymbol{\gamma}$ and empirical categorical distributions. (We use this definition of $\hat{\boldsymbol{\gamma}}$ in the proof of Theorem 1 in Appendix F.) To sample from the revised distribution $p_\gamma(t_m|\mathbf{x},\mathbf{z};l_m)$ we first draw a Bernoulli sample $\hat{\lambda}_m \sim \text{Bern}(\hat{\lambda}_m; 1/(1+\lambda l_m))$ and consequently we draw knowledge sample from the mixture of probabilities by $t_m \sim \hat{\lambda}_m r_\gamma(t_m|z_m) + (1-\hat{\lambda}_m)p_\mathcal{D}(t_m|x_m)$. We implement this with a single-sample approximation.

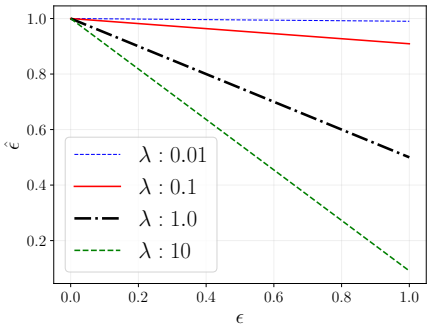


Figure 5: Effect of λ and observation probability ϵ on $\hat{\epsilon}$

E Revision Step Property of $p_\gamma(t_m|x_m,z_m;l_m)$

In this section we show how the formulation of $p_\gamma(t_m|x_m,z_m;l_m)$ leads to the property that, during the revision step, we rely more on the available side/external knowledge as more of it becomes available. Recall that we initially defined $p_\gamma(t_m|x_m,z_m;l_m) = \frac{1}{1+\lambda l_m}r_\gamma(t_m|z_m) + \frac{\lambda l_m}{1+\lambda l_m}p_\mathcal{D}(t_m|x_m)$, where λ allows a fractional portion of the empirical distribution $p_\mathcal{D}$ to influence the learned model. Suppose that $l_m \sim \text{Bern}(\epsilon)$ and $\tilde{\lambda} = 1/(1+\lambda)$, given node m we have

$$\begin{aligned}
p(t_m|x_m,z_m) &= \sum_{l_m} p(t_m|x_m,z_m,l_m)p(l_m) \\
&= \epsilon p_\gamma(t_m|x_m,z_m,l_m) + (1-\epsilon)r_\gamma(t_m|z_m) \\
&= \epsilon\Big[\tilde{\lambda}r_\gamma(t_m|z_m) + (1-\tilde{\lambda})p_\mathcal{D}(t_m|x_m)\Big] \\
&\quad + (1-\epsilon)r_\gamma(t_m|z_m) \\
&= \big(1-\epsilon(1-\tilde{\lambda})\big)r_\gamma(t_m|z_m) \\
&\quad + \epsilon(1-\tilde{\lambda})p_\mathcal{D}(t_m|x_m) \\
&= \hat{\epsilon}r_\gamma(t_m|z_m) + (1-\hat{\epsilon})p_\mathcal{D}(t_m|x_m),
\end{aligned}$$

where $\hat{\epsilon} = 1 - \epsilon\big(\frac{\lambda}{1+\lambda}\big)$. In particular, this formulation allows us to account for both our estimate of observation of side knowledge, and the extent to which we wish to use it when learning our model. In Fig. 5, we show the effect of λ and ϵ on $\hat{\epsilon}$.

F Generalization of SSDVAE (Proof of Theorem 1)

We show that under specific setting, our model reduces to the SSDVAE parameter injection mechanism.

Theorem. *If $r_\gamma(t|z) = Cat(\boldsymbol{\gamma})$, a categorical distribution with parameters γ, and the empirical dis-*

tribution $p_{\mathcal{D}}(t|x)$ is a one-hot representation with $t_{k^*} = 1$, the revision step reduces to SSDVAE parameter injection.

Proof. Using the interpolation form $p_\gamma\big(t_m|x_m, z_m; l_m\big) = \hat{\lambda}_m r_\gamma(t_m|z_m) + (1 - \hat{\lambda}_m)p_{\mathcal{D}}(t_m|x_m)$ and due to the fact that the mixture of categorical distributions is a single categorical, we have $p\big(t|x, z; l = 1\big) = \mathrm{Cat}(\hat{\boldsymbol{\gamma}})$, where

$$\hat{\boldsymbol{\gamma}} = [\hat{\lambda}\gamma_1, \hat{\lambda}\gamma_2, \ldots, \hat{\lambda}\gamma_{k^*} + (1 - \hat{\lambda}), \ldots, \hat{\lambda}\gamma_K]. \tag{35}$$

Given $0 \leq \gamma_i \leq 1$ and $0 \leq \hat{\lambda} \leq 1$, we have $\hat{\lambda}\gamma_i \leq \gamma_i$ $(\forall i \neq k^*)$, and $\hat{\lambda}\gamma_{k^*} + (1 - \hat{\lambda}) \geq \gamma_{k^*}$. This implies that for all the indices $i \neq k^*$, the value of γ_i is reduced and γ_{k^*} is increased, confirming that parameter injection is a specific form of our definition. □

G Updating Step Maximizes Mutual Information (Proof of Theorem 2)

We show that for the reconstruction tasks, where $\mathbf{y}$ is a copy of $\mathbf{x}$, minimizing $\mathcal{I}(\mathbf{x}; \mathbf{t}|\mathbf{z})$ maximizes $\mathcal{I}(\mathbf{t}; \mathbf{z})$. While not explicitly reflected in our model definition, note that $\mathcal{I}(\mathbf{t}; \mathbf{z})$ is intractable. This can be seen in the definition of $\mathcal{I}(\mathbf{t}; \mathbf{z})$,

$$\mathcal{I}(\mathbf{t}; \mathbf{z}) = \mathbb{E}_{p(\mathbf{t},\mathbf{z})} \log \frac{p(\mathbf{t}|\mathbf{z})}{p(\mathbf{t})}, \tag{36}$$

here both $p(\mathbf{t}, \mathbf{z})$ and $p(\mathbf{t})$ are intractable due to the integral over $\mathbf{x}$. Following Cover (1999), the mutual information between three random variables $\mathbf{x}$, $\mathbf{t}$, and $\mathbf{z}$ can be defined as

$$\mathcal{I}(\mathbf{x}; \mathbf{t}; \mathbf{z}) = \mathcal{I}(\mathbf{x}; \mathbf{t}) - \mathcal{I}(\mathbf{x}; \mathbf{t}|\mathbf{z}). \tag{37}$$

Eq. 37 is symmetric in $\mathbf{x}$, $\mathbf{t}$ and $\mathbf{z}$, so we have:

$$\mathcal{I}(\mathbf{x}; \mathbf{t}; \mathbf{z}) = \mathcal{I}(\mathbf{t}; \mathbf{z}) - \mathcal{I}(\mathbf{t}; \mathbf{z}|\mathbf{x}). \tag{38}$$

As already proven by Federici et al. (2019), the second term $\mathcal{I}(\mathbf{t}; \mathbf{z}|\mathbf{x})$ equals to zero. Combining Eq. 37 and Eq. 38, we have

$$\mathcal{I}(\mathbf{x}; \mathbf{t}) - \mathcal{I}(\mathbf{x}; \mathbf{t}|\mathbf{z}) = \mathcal{I}(\mathbf{t}; \mathbf{z}) \tag{39}$$

As evident from Eq. 39, maximizing $\mathcal{I}(\mathbf{x}; \mathbf{t})$ and minimizing $\mathcal{I}(\mathbf{x}; \mathbf{t}|\mathbf{z})$ is equivalent to maximizing $\mathcal{I}(\mathbf{t}; \mathbf{z})$.

H Full Results

In this section, we present the full results for those presented in the main paper. Specifically, the results presented here augment those in Fig. 4, Table 1, and Table 2.

Results on the Effect of Noisy Knowledge (η) In Table 3, we how the full results from our noisy knowledge experiments, shown in the main paper's Table 1. As with our other results, these are averaged over three runs. In our development we noticed that precision tended to be the main hindrance on F1, and therefore we additionally report precision.

Full Perplexity Results In Table 4, we how the full language modeling perplexity results that are shown in the main paper's Table 2. Here, we are showing results across both the dev and test sets. We also include standard deviation results (from three runs).

Full Results on Frame Classification In Table 5, we show the full results for the side knowledge (per-event frame) prediction experiments we reported in Fig. 4. The main classification results were averaged over three runs, along with standard deviation. These results are visible in Fig. 4, but we include the numeric values for any future comparisons.

I Setup/Implementation Details

The NYT dataset is available through the LDC and a research license, and the Wikipedia dataset is publicly available under a CC BY-NC 4.0 license. For both datasets, the token vocabulary size is 150k and the number of unique semantic frames is 600. In line with previous research (Rezaee and Ferraro, 2021), for the documents with more than 5 events, we used the first 5 events that had frames. Each model was trained using a single GPU (an RTX 2080 TI, TITAN RTX, or a Quadro 8000). Each RevUp model took approximately 12 hours to train.

We noticed that the observability of side knowledge can affect the relative importance of the loss terms in Eq. 10. Given a mini batch of training data, we define $\hat{\epsilon}$ as an approximation of ϵ. To do so, we calculate the number of observed frames over total frames. In all cases, high importance placed on the regularization term $\mathcal{L}_t$ (Eq. 9) was important to help prevent overfitting. When frames were frequently observed ($\hat{\epsilon} \geq 0.5$), the model needed to rely more heavily on the updating phase ($\mathcal{L}_\mathcal{I}$, Eq. 4) and the regularization on z ($\mathcal{L}_z$, Eq. 7) was less important; specifically, we found $\alpha = 0.3$, $\beta = 1\mathrm{e}-6$ and $\zeta = 0.7$ to be effective. Otherwise, we found the model needed to place small but relatively equal weight on the updating phase and z regularization components; specifically, we found

model	noise η	Dataset					
		Wiki			NYT		
		Acc	F1	Prec	Acc	F1	Prec
SSDVAE	0.1	0.77 ± 0.008	0.43 ± 0.019	0.41 ± 0.015	0.76 ± 0.000	0.55 ± 0.001	0.52 ± 0.001
RevUp		$\mathbf{0.85 \pm 0.005}$	$\mathbf{0.58 \pm 0.015}$	$\mathbf{0.56 \pm 0.012}$	$\mathbf{0.83 \pm 0.000}$	$\mathbf{0.71 \pm 0.002}$	$\mathbf{0.69 \pm 0.002}$
SSDVAE	0.2	0.69 ± 0.002	0.35 ± 0.001	0.33 ± 0.002	0.67 ± 0.001	0.44 ± 0.000	0.41 ± 0.001
RevUp		$\mathbf{0.82 \pm 0.001}$	$\mathbf{0.52 \pm 0.001}$	$\mathbf{0.49 \pm 0.003}$	$\mathbf{0.80 \pm 0.000}$	$\mathbf{0.63 \pm 0.001}$	$\mathbf{0.60 \pm 0.001}$
SSDVAE	0.3	0.60 ± 0.010	0.28 ± 0.018	0.27 ± 0.013	0.58 ± 0.001	0.36 ± 0.000	0.34 ± 0.000
RevUp		$\mathbf{0.79 \pm 0.008}$	$\mathbf{0.45 \pm 0.023}$	$\mathbf{0.42 \pm 0.019}$	$\mathbf{0.77 \pm 0.000}$	$\mathbf{0.58 \pm 0.001}$	$\mathbf{0.55 \pm 0.000}$
SSDVAE	0.5	0.41 ± 0.017	0.17 ± 0.014	0.18 ± 0.011	0.41 ± 0.003	0.23 ± 0.001	0.23 ± 0.002
RevUp		$\mathbf{0.72 \pm 0.007}$	$\mathbf{0.36 \pm 0.022}$	$\mathbf{0.34 \pm 0.017}$	$\mathbf{0.71 \pm 0.001}$	$\mathbf{0.48 \pm 0.001}$	$\mathbf{0.45 \pm 0.000}$
SSDVAE	0.7	0.23 ± 0.002	0.09 ± 0.001	0.11 ± 0.001	0.22 ± 0.004	0.11 ± 0.002	0.13 ± 0.002
RevUp		$\mathbf{0.64 \pm 0.005}$	$\mathbf{0.29 \pm 0.003}$	$\mathbf{0.28 \pm 0.002}$	$\mathbf{0.63 \pm 0.003}$	$\mathbf{0.37 \pm 0.002}$	$\mathbf{0.35 \pm 0.002}$
SSDVAE	0.9	0.02 ± 0.001	0.00 ± 0.000	0.01 ± 0.000	0.02 ± 0.001	0.00 ± 0.000	0.01 ± 0.000
RevUp		$\mathbf{0.41 \pm 0.009}$	$\mathbf{0.09 \pm 0.003}$	$\mathbf{0.11 \pm 0.003}$	$\mathbf{0.25 \pm 0.039}$	$\mathbf{0.06 \pm 0.013}$	$\mathbf{0.09 \pm 0.015}$

Table 3: Effect of Noise on the test dataset.

model	ϵ	Dataset			
		Wiki		NYT	
		Valid	Test	Valid	Test
RNNLM	-	64.02 ±2.53	64.57 ±2.60	65.07 ±3.25	56.96 ±2.82
HAQAE	-	49.03 ±3.90	50.10 ±4.05	43.13 ±5.29	39.47 ±4.84
SSDVAE	0.0	46.61 ±1.08	47.50 ±1.06	44.80 ±0.85	39.75 ±1.21
RevUp		**44.38** ±1.38	**45.36** ±1.40	**42.61** ±0.58	**39.48** ±0.49
SSDVAE	0.1	45.17 ±1.17	45.91 ±1.15	43.52 ±0.16	39.73 ±0.16
RevUp		**43.99** ±1.79	**44.87** ±1.83	**36.10** ±0.94	**33.34** ±0.96
SSDVAE	0.7	44.38 ±0.55	44.79 ±0.58	39.77 ±1.05	36.79 ±0.91
RevUp		**40.90** ±1.11	**41.74** ±1.05	**35.77** ±0.25	**33.20** ±0.17
SSDVAE	1.0	36.79 ±0.33	36.96 ±0.34	33.18 ±0.39	30.69 ±0.31
RevUp		**34.28** ±0.89	**34.85** ±0.90	**30.61** ±0.38	**28.33** ±0.43

Table 4: Perplexity results on the Wikipedia and NYT datasets (extension of Table 2). Main results are the average of three runs, along with standard deviation.

$\alpha = 0.1$, $\beta = 0.2$ and $\zeta = 1.0$ to be effective. We obtained these values via experimentation on dev data.

Following SSDVAE and HAQAE models, we use pre-trained Glove 300 embeddings to represent tokens and used gradient clipping at 5.0 to prevent exploding gradients. We use a two-layer of bi-directional GRU for the encoder and a two-layer uni-directional GRU for the decoder (with a hidden dimension of 512 for both). We used early stopping (lack of validation performance improvement for 3 iterations).

model	ϵ	Dataset					
		Wiki			NYT		
		Acc	F1	Prec	Acc	F1	Prec
SSDVAE	0.1	0.02 ± 0.002	0.01 ± 0.001	0.02 ± 0.002	0.02 ± 0.002	0.00 ± 0.001	0.01 ± 0.003
RevUp		$\mathbf{0.06 \pm 0.001}$	$\mathbf{0.02 \pm 0.000}$	$\mathbf{0.04 \pm 0.000}$	$\mathbf{0.05 \pm 0.003}$	$\mathbf{0.02 \pm 0.001}$	$\mathbf{0.04 \pm 0.002}$
SSDVAE	0.3	0.10 ± 0.002	0.05 ± 0.001	0.08 ± 0.002	0.11 ± 0.003	0.06 ± 0.001	0.09 ± 0.002
RevUp		$\mathbf{0.23 \pm 0.002}$	$\mathbf{0.11 \pm 0.001}$	$\mathbf{0.13 \pm 0.000}$	$\mathbf{0.17 \pm 0.005}$	$\mathbf{0.07 \pm 0.002}$	$\mathbf{0.09 \pm 0.002}$
SSDVAE	0.5	0.23 ± 0.002	0.11 ± 0.000	0.14 ± 0.001	0.25 ± 0.005	0.15 ± 0.002	0.18 ± 0.003
RevUp		$\mathbf{0.44 \pm 0.014}$	$\mathbf{0.22 \pm 0.009}$	$\mathbf{0.21 \pm 0.006}$	$\mathbf{0.36 \pm 0.017}$	$\mathbf{0.22 \pm 0.008}$	$\mathbf{0.22 \pm 0.007}$
SSDVAE	0.7	0.52 ± 0.008	0.28 ± 0.008	0.28 ± 0.007	0.56 ± 0.005	0.40 ± 0.005	0.39 ± 0.004
RevUp		$\mathbf{0.84 \pm 0.004}$	$\mathbf{0.65 \pm 0.003}$	$\mathbf{0.65 \pm 0.002}$	$\mathbf{0.76 \pm 0.010}$	$\mathbf{0.61 \pm 0.015}$	$\mathbf{0.57 \pm 0.018}$
SSDVAE	0.9	0.84 ± 0.004	0.61 ± 0.008	0.62 ± 0.010	0.83 ± 0.001	0.77 ± 0.013	0.78 ± 0.014
RevUp		$\mathbf{0.86 \pm 0.001}$	$\mathbf{0.72 \pm 0.003}$	$\mathbf{0.73 \pm 0.005}$	$\mathbf{0.85 \pm 0.002}$	$\mathbf{0.81 \pm 0.007}$	$\mathbf{0.82 \pm 0.008}$
SSDVAE	1.0	0.85 ± 0.002	0.68 ± 0.003	0.69 ± 0.003	0.84 ± 0.001	0.81 ± 0.002	0.82 ± 0.001
RevUp		$\mathbf{0.87 \pm 0.003}$	$\mathbf{0.75 \pm 0.008}$	$\mathbf{0.76 \pm 0.009}$	$\mathbf{0.85 \pm 0.001}$	$\mathbf{0.82 \pm 0.007}$	$\mathbf{0.83 \pm 0.006}$

Table 5: Classification Results: these are the full numeric values (averages and standard deviations computed from three runs) for the graphs shown in Fig. 4.