Hidden Schema Networks

Ramsés J. Sánchez[1,2*], Lukas Conrads[1,2], Pascal Welke[1,2,3],
Kostadin Cvejoski[1,4], César Ojeda[5]

[1]Lamarr Institute, [2]University of Bonn, [3]TU Wien, [4]Fraunhofer IAIS
[5] University of Potsdam

Abstract

Large, pretrained language models infer powerful representations that encode rich semantic and syntactic content, albeit implicitly. In this work we introduce a novel neural language model that enforces, via inductive biases, explicit relational structures which allow for compositionality onto the output representations of pretrained language models. Specifically, the model encodes sentences into sequences of symbols (composed representations), which correspond to the nodes visited by biased random walkers on a global latent graph, and infers the posterior distribution of the latter. We first demonstrate that the model is able to uncover ground-truth graphs from artificially generated datasets of random token sequences. Next, we leverage pretrained BERT and GPT-2 language models as encoder and decoder, respectively, to infer networks of symbols (schemata) from natural language datasets. Our experiments show that (i) the inferred symbols can be interpreted as encoding different aspects of language, as e.g. topics or sentiments, and that (ii) GPT-like models can effectively be conditioned on symbolic representations. Finally, we explore training autoregressive, random walk "reasoning" models on schema networks inferred from commonsense knowledge databases, and using the sampled paths to enhance the performance of pretrained language models on commonsense *If-Then* reasoning tasks.

1 Introduction

Much of the developmental and causal theories of human cognition are predicated on *relational* structures of knowledge that naturally exhibit *compositionality*. Semantic content is intrinsically relational, as one is only able to explain a given unit of knowledge – such as a concept, word or perception – insofar as there are other units of knowledge which relate to it (Block, 1986; Margolis and Laurence, 1999). Thus we can partially construe a concept through its relationships to other concepts (like when we say "a dog is an animal that barks"), just as we can partially construe it through its relationships to our perceptions (when we say "*that* is a dog", whilst pointing to a dog on the street) or the words we use (when we use the word *dog* to refer to the concept *dog*). Likewise, we can partially construe words not only through their relationships to concepts or percepts, but also through their relationships to other words, as words that occur in the same context tend to have similar meanings (Harris, 1954; Firth, 1957). Note that is precisely this contextual semantic word content what we explicit have access to, when processing our raw text datasets. On the other hand, generalization, reasoning and understanding seem to be inevitably tied to the compositional nature of knowledge. Indeed, the ability to compose a set of knowledge units (and their relations) into new, more complex relational units, which can be deployed to understand and reason about unseen data – a feature usually referred to as combinatorial generalization – is regarded as key to human-level intelligence (Fodor and Pylyshyn, 1988; Fodor and Lepore, 2002; Lake et al., 2017; Battaglia et al., 2018). Relational structures allowing for compositionality thus seem to comprise not a sufficient, but a necessary attribute of any representation scheme that strives for the generalization power of human cognition.

From the computational side, if one is to inform any modern language model with such structural characteristics, one will initially encounter the problem of finding suitable primitives or data structures. Distributed continuous word representations (Bengio et al., 2003), for example, are routinely leveraged in many different downstream tasks. These representations are trained to encode average contextual semantics – precisely the kind of semantic content typical of word co-occurrence relations we mentioned above – into a semantic space, which allows meaning to change continuously within it (Mikolov et al., 2013). Yet, de-

spite earlier attempts (Mitchell and Lapata, 2008), it is unclear whether such representations can be meaningfully composed into representations of, say, unseen sentences and thus mimic the compositional character of natural language. More recently, contextualized continuous word representations inferred by deep learning architectures have shown spectacular results in many NLP tasks (Radford et al., 2018; Devlin et al., 2018; Radford et al., 2019; Brown et al., 2020). Their success stems from those models' ability to infer flexible representations through, *inter alia*, raw, massive datasets, data-scalable attention mechanisms and minimal inductive biases (Vaswani et al., 2017). These representations are known to not only contain rich contextual word semantics, but also consistently encode sentence-level grammar (Hewitt and Manning, 2019), and the models from which they are obtained have been shown to implement different notions of compositionality too (Hupkes et al., 2020; Wei et al., 2022). Nevertheless, it is still unclear whether such representations can be composed into representations of novel sentences (Yu and Ettinger, 2020; Bhathena et al., 2020). In fact, most of their syntactic properties are implicit and therefore inferred only a posteriori, typically through probes which neither guarantee their presence, nor establish how they were obtained in the first place (Liu et al., 2019; Rogers et al., 2020; Ravichander et al., 2021).

Roughly following the program outlined by Tenenbaum et al. (2011), we develop a novel, neural language model – the Hidden Schema Network model (HSN) – that enforces, via inductive biases, a discrete, *relational* structure for sentence representation which allows for *compositionality*[1] onto the output representations of large, pretrained language models (LPLM). Using a variational autoencoder (VAE) framework (Kingma and Welling, 2013; Rezende et al., 2014), HSN leverages LPLMs to encode natural language sentences into sequences of symbols, which correspond to the nodes visited by biased random walkers on a global latent graph, while inferring the posterior distribution of the latter.

In practice, translating the implicit knowledge encoded by LPLMs into explicit relational struc-

tures has some naturally appealing properties. For example, HSN can support symbolic reasoning via inference of missing semantic connections, through high-order paths along the inferred graphs (Lao et al., 2011; Das et al., 2018; Chen et al., 2020). Likewise secondary, autoregressive "reasoning" models can be trained on the symbol sequences inferred by HSN from task-specific natural language sequences, like e.g. question-answer pairs. The reasoning paths sampled from such autoregressive models could then be used to inform GPT-like models and improve their performance on natural language understanding tasks, with which they are known to struggle (Brown et al., 2020; Dunietz et al., 2020; Talmor et al., 2021). Below we demonstrate that (i) HSN is able to uncover ground-truth graphs from artificially generated datasets of random token sequences, and that (ii) using pretrained BERT and GPT-2 language models as encoder and decoder, respectively, HSN is able to infer schema networks from natural language datasets, whose symbols encode different aspects of language (like e.g. topics or sentiments). Finally, we also explore training secondary, autoregressive models on the schema networks inferred from commonsense knowledge databases, and using the sampled paths to enhance the performance of LPLM on commonsense *If-Then* reasoning tasks.

2 Related Work

In cognitive psychology, a schema is (roughly) defined as a large, complex unit of knowledge representing what is typical of a group of instances (Bartlett, 1932; Piaget, 1948; Rumelhart, 2017). Marvin Minsky's frames (Minsky, 1974, 1975) are similar in function to a schema, but perhaps more easily characterized in terms of data structures. We use these terms in a loose fashion, however. Our aim being only to be suggestive of the general problem of knowledge representation (Thagard, 1984). We are in fact concerned with representation schemes for natural language processing. Within the context of linguistics, Jackendoff (1978) argues that there must be a level of representation – the so-called conceptual structures – at which information conveyed by language must be compatible with information coming from sensory systems. Conceptual structures must, he goes on, be able to represent all the conceptual distinctions made by natural language, and provide some degree of compositionality. Earlier computational models

---

[1]Note that throughout the paper we refer only to *compositionality of representations* and not to the compositional functions that can be implemented by the models we use. The latter, functional compositionality, is studied by e.g. Hupkes et al. (2020).
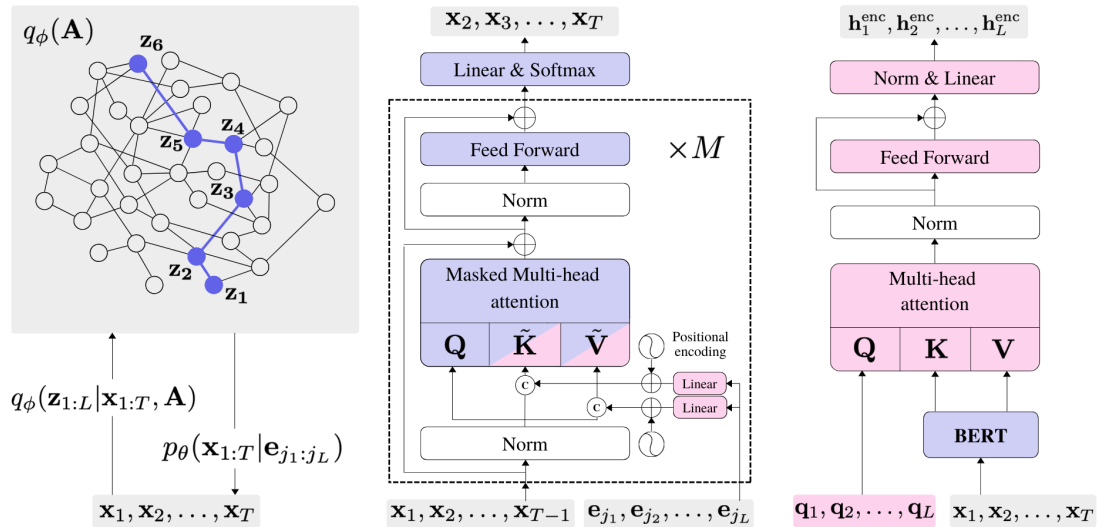
Figure 1: **Left:** Hidden Schema Network model. **Center:** Decoder architecture as a modified GPT-2 with M layers and pseudo-self-attention mechanism to attend to the schema $\mathbf{e}_{j_1:j_L}$. This schema is defined as a sequence of L integers, each of which is represented as a one-hot vector. The "c" operations labels concatenation. **Right:** Encoder architecture as BERT, followed by a single Transformer block. In both center and right figure purple (pink) shaded blocks represent submodules with pretrained (randomly initialized) parameters.

implementing (some kind of) conceptual structure rely on either hand-coded (semantic) network representations (Quillan, 1966; Collins and Quillian, 1969; Brachman, 1977) or hand-coded databases (McClelland and Rogers, 2003). Other works focus instead on learning semantic representations from raw text data directly, via topic models (Griffiths et al., 2007b), and even infer latent concept graphs through nonparametric priors (Chambers et al., 2010). In sharp contrast with these works, (a line of) modern, neuro-symbolic reasoning approaches in NLP seek to combine the implicit knowledge of LPLM with external information, structured as knowledge graphs, in order to solve commonsense question-answering tasks (Lin et al., 2019; Ma et al., 2019; Yasunaga et al., 2021; Verga et al., 2021; Kapanipathi et al., 2021; Hu et al., 2022b). Note that the knowledge graphs here can also be interpreted as (some class of) human-authored conceptual structure (see however West et al. (2021), who shows that LPLM can generate commonsense knowledge graphs too). Our work runs perpendicular to these approaches, for it leverages LPLMs to automatically discover, in an unsupervised fashion, knowledge graphs in representation space, and introduces reasoning via random walk processes conditioned on the discovered graphs. This problem necessarily involves the (neural) inference of discrete variables, which has also been successfully treated in the past (van den Oord et al., 2017; Hu

et al., 2017; Zhao et al., 2018; Kaiser and Bengio, 2018; Kaiser et al., 2018; Stratos and Wiseman, 2020).

3 Hidden Schema Networks

We address the problem of learning the joint probability distribution over sequences of words, while inferring symbolic representations capturing their semantics. Neural autoregressive language models approximate such distributions with a product over conditional probabilities, such that

$$p(\mathbf{x}_{1:T}) = \prod_{i=1}^{T} p_\theta(\mathbf{x}_i|\mathbf{x}_{<i}), \qquad (1)$$

where $\mathbf{x}_{1:T} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ labels the sequence of words in question, and each conditional is given by (the pdf of) a categorical distribution over some vocabulary of size V. The class probabilities of these conditionals are generally computed as $\boldsymbol{\pi}_i = \mathrm{softmax}(\mathbf{W} \cdot \mathbf{h}_\theta(\mathbf{x}_{<i}))$, with $\mathbf{W} \in \mathbb{R}^{V \times D}$ trainable, and D the output dimension of $\mathbf{h}_\theta$, a deep neural network model with parameter set θ (Bengio et al., 2003). Models of this form allow for tractable estimation of and sampling from either the joint distribution, or any product of the conditionals in Eq. 1. Indeed, their recent implementation in terms of large-capacity, self-attention architectures, the likes of GPT-3 (Brown et al., 2020), has been shown to generate syntactically correct, diverse and

fluent text. In what follows we attempt to condition the joint distribution of Eq. 1 on an additional latent, discrete representation which can, at least in principle, capture the relational and compositional features of semantic content.

Let us assume there is a set of K symbols, $\mathcal{E} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_K\}$, each of which encode some high-level, abstract semantic content of natural language. Let this set be the set of nodes of a hidden (semantic) graph $\mathcal{G}$, with adjacency matrix $\mathbf{A}$, so that adjacent (connected) symbols are semantically related. These symbols can generically be defined as learnable, dense vectors in $\mathbb{R}^S$, for some dimension S. Without loss of generality, however, we opt below for simple indicator ("one-hot") vectors of dimension K instead. We define a *schema* $\mathbf{e}_{j_1:j_L}$ as a sequence of $L \ll K$ symbols $(\mathbf{e}_{j_1}, \mathbf{e}_{j_2}, \ldots, \mathbf{e}_{j_L})$, where the indices $j_1, \ldots, j_L$ label a subset of connected nodes in $\mathcal{G}$. Accordingly, we refer to $\mathcal{G}$ as a *schema network*. The symbols composing the schemata are chosen through a L-step stochastic process conditioned on $\mathcal{G}$. Partially motivated by research on random walks and human memory search (Griffiths et al., 2007a; Abbott et al., 2012), as well as by the simplicity of their inference, we choose to compose the schemata via biased random walk processes on $\mathcal{G}$, and leave exploring different schema processes for future work. Finally, let us assume that natural language sentences are generated conditioned on these schema representations. We can then define a generative language model as follows.

3.1 Generative Model

Let the joint probability over a sequence $\mathbf{x}_{1:T}$ of T words, together with the hidden graph $\mathcal{G}$, be

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{A}) = \sum_{\mathbf{z}_{1:L}} p_\theta(\mathbf{x}_{1:T}|\mathbf{e}_{j_1:j_L}) p(\mathbf{z}_{1:L}|\mathbf{A}) p(\mathbf{A}), \quad (2)$$

where $\mathbf{z}_{1:L}$ labels the sequence $\mathbf{z}_1, \ldots, \mathbf{z}_L$ of K-dimensional, one-hot vectors representing the node labels $j_1, \ldots, j_L$ visited by a random walker on $\mathcal{G}$, and θ denotes the trainable model parameters. Note that we introduced the one-hot representation of j_i for notational convenience, as shall become evident below[2]. Next, we specify the different components of Eq. 2.

[2] Explicitly, j_i denotes the index of the non-zero component of $\mathbf{z}_i$, i.e. $j_i = \{k \in [1, K] : z_i^k = 1\}$, with the superindex k denoting the components of $\mathbf{z}_i$.

Prior over (global) graph. A prior on the adjacency matrix $p(\mathbf{A})$ allows us to control the topological properties of $\mathcal{G}$. One can choose, for example, random graph models whose degree distributions asymptotically follow a power law (Barabási and Albert, 1999), or unbiased, maximum entropy graph models, with respect to some given constrains (Park and Newman, 2004). Here we choose a Bernoulli (Erdös-Rényi) random graph model (Solomonoff and Rapoport, 1951; Erdös and Rényi, 1959), for which each link is defined via an independent Bernoulli variable with some fixed, global probability $p \in [0, 1]$, which is a hyperparameter of our model.

Prior over random walks. The probability $p(\mathbf{z}_{1:L}|\mathbf{A})$ of a random walk over the nodes of $\mathcal{G}$ can generally be written as

$$p(\mathbf{z}_{1:L}|\mathbf{A}) = p(\mathbf{z}_1) \prod_{i=2}^{L} p(\mathbf{z}_i|\mathbf{z}_{i-1}, \mathbf{A})$$

$$= \left(\prod_{m=1}^{K} \rho_m^{z_1^m} \right) \prod_{i=2}^{L} \left(\prod_{j=1}^{K} \prod_{k=1}^{K} P_{k,j}^{z_i^k z_{i-1}^j} \right), \quad (3)$$

where $p(\mathbf{z}_1)$ labels the probability of selecting j_1 as the starting point of the walk, and it is given by (the pdf of) a categorical distribution over the nodes of $\mathcal{G}$, with class probabilities $\{\rho_i\}_{i=1}^{K}$. Similarly $p(\mathbf{z}_i|\mathbf{z}_{i-1}, \mathbf{A})$ labels the conditional probability of jumping from j_{i-1} to j_i, which we define in terms of a $K \times K$ transition probability matrix $\mathbf{P}$. Now, to allow for biased random walks, let each node k on $\mathcal{G}$ be given a positive weight f_k, so that the probability of jumping from j to k is proportional to $f_k A_{kj}$. We then write the transition probability matrix as

$$P_{k,j} = \frac{f_k A_{kj}}{\sum_{i=1}^{K} f_i A_{ij}}, \quad (4)$$

so that the motion of the random walker is biased according to the node weights f_k. These weights should be understood as encoding aspects of the diffusion dynamics that are independent of the topology of the graph (Gómez-Gardenes and Latora, 2008; Lambiotte et al., 2011). Three comments are in order: first, note that one can also *train* the prior over walks by making the vectors $\boldsymbol{\rho}$ and $\mathbf{f}$ learnable. Second, setting the node weights $\mathbf{f} = \mathbb{I}$ and the class probabilities $\boldsymbol{\rho} = \frac{1}{K}\mathbb{I}$, with $\mathbb{I}$ the K-dimensional vector of ones, yields a *uniform* random walk over $\mathcal{G}$, i.e. a process in which the walker has equal probability of jumping to any of

its neighbors. Third, one can also allow for *inhomogeneous* random walks in which the probability matrix changes at each step of the random walk. Such processes can be parameterized with a sequence of weights $\mathbf{f}^{[1]}, \mathbf{f}^{[2]}, \ldots, \mathbf{f}^{[L-1]}$.

Decoder and likelihood. Just as in Eq. 1 we define the joint probability over word sequences as a product of conditional probabilities, each of which is now conditioned on the schema $\mathbf{e}_{j_1:j_L}$ too. Accordingly, the class probabilities of the ith conditional are now given by $\boldsymbol{\pi}_i = \text{softmax}(\mathbf{W} \cdot \mathbf{h}_\theta^{\text{dec}}(\mathbf{x}_{<i}, \mathbf{e}_{j_1:j_L}))$, with $\mathbf{W} \in \mathbb{R}^{V \times D}$ trainable and $\mathbf{h}_\theta^{\text{dec}}$ a deep neural network model. We let $\mathbf{h}_\theta^{\text{dec}}$ be a *pretrained* GPT-2 language model, and modify it to also process the schema $\mathbf{e}_{j_1:j_L}$, but remark that any other model for sequence processing (as e.g. a recurrent neural net) could be used instead. A bit more in detail, to condition GPT-2 on $\mathbf{e}_{j_1:j_L}$, without perturbing its optimized weights too much, we use the *pseudo-self*-attention (PSA) mechanism introduced by (Ziegler et al., 2019). In a nutshell, this mechanism augments the key and value matrices of GPT-2 in their first L rows with projections of $\mathbf{e}_{j_1:j_L}$. Figure 1 shows an illustration of the complete decoder model, including the PSA mechanism. Please check Appendix A for the explicit equations of the latter.

3.2 Inference Model

The generative model we presented above is hierarchical. The random graph is shared across all sentences and thus constitutes a global latent object. The random walks, in contrast, are local random variables. Our task is to infer the schema and graph posterior distributions that best describe the collection of word sequences in our datasets. To do this, we approximate the true posterior distribution of these variables with a variational posterior of the form

$$q_\phi(\mathbf{z}_{1:L}, \mathbf{A} | \mathbf{x}_{1:T}) = q_\phi(\mathbf{z}_{1:L} | \mathbf{x}_{1:T}, \mathbf{A}) q_\phi(\mathbf{A}), \quad (5)$$

where ϕ labels the set of trainable parameters. We model $q_\phi(\mathbf{A})$, the posterior over the graph, assigning again Bernoulli variables to its links, but we let the probability of observing each link depend on the global symbols. That is, we model the link probabilities as $p_\phi(\mathbf{e}_i, \mathbf{e}_j) = \text{sigmoid}(g_\phi(\mathbf{e}_i, \mathbf{e}_j))$, for all $\mathbf{e}_i \in \mathcal{E}$, with $g_\phi : \mathcal{E} \times \mathcal{E} \to \mathbb{R}$ a deep neural network.

Likewise we model $q_\phi(\mathbf{z}_{1:L} | \mathbf{x}_{1:T}, \mathbf{A})$, the posterior probability over random walks on the graph,

analog to Eq. 3. This time, however, instead of having a single transition probability matrix, we have a sequence of them, thereby allowing the posterior to capture inhomogeneous random walks. We therefore model the class probabilities over the starting point of the random walks as $\boldsymbol{\rho}(\mathbf{x}_{1:T}, \phi) = \text{softmax}(\mathbf{h}_1^{\text{enc}})$, whereas the sequence of transition matrices is given by

$$Q_{k,j}^{[i]}(\mathbf{x}_{1:T}, \mathbf{A}, \phi) = \frac{f_k^{[i]}(\mathbf{x}_{1:T}, \phi) \, A_{kj}}{\sum_m f_m^{[i]}(\mathbf{x}_{1:T}, \phi) \, A_{mj}}, \quad (6)$$

with the sequence of node weights $\mathbf{f}^{[1]}, \mathbf{f}^{[2]} \ldots, \mathbf{f}^{[L-1]} = \exp(\mathbf{h}_{2:L}^{\text{enc}})$. Now, the set $\mathbf{h}_1^{\text{enc}}, \mathbf{h}_2^{\text{enc}}, \ldots, \mathbf{h}_L^{\text{enc}} \in \mathbb{R}^K$ is the sequence of outputs of a deep neural network model $\mathbf{h}_\phi^{\text{enc}}(\mathbf{x}_{1:T})$ processing the input sequence of T words. The model $\mathbf{h}_\phi^{\text{enc}}(\mathbf{x}_{1:T})$ must then map a sequence of T vectors to a sequence of L vectors. We define $\mathbf{h}_\phi^{\text{enc}}$ by a *pretrained* BERT model (Devlin et al., 2018), followed by a single Transformer block, randomly initialized. The Transformer block processes the T (D-dimensional) outputs from BERT as keys and values, together with a set of L learnable vectors $\mathbf{q}_{1:L}$ as queries. The right-hand side of Figure 1 illustrates the complete encoder architecture. For completeness, we also give the explicit and complete expressions of both posterior graph and random walk models in Appendix C.

3.3 Training Objective

To optimize the parameter sets $\{\theta, \phi\}$ of our latent variable model we would, as usual, maximize a variational lower bound on the logarithm of the marginal likelihood $p_\theta(\mathbf{x}_{1:T})$ (Bishop, 2006). It is, however, well known that VAE models tend to encounter problems learning representations encoding information about the data – the so-called posterior collapse problem – especially when dealing with natural language (Bowman et al., 2015). To solve this issue practitioners resort to maximizing the variational lower bound, together with the mutual information between data and representations (Zhao et al., 2018; Fang et al., 2019; Zhao et al., 2019). We follow this same route here and refer the reader to Appendix D for the derivation of our objective function. The training algorithm is presented in Appendix B and training times in Appendix I. Source code to reproduce all experiments is available online.[3]

---

[3]Source code: https://github.com/ramsesjsf/HiddenSchemaNetworks

Graph $\mathcal{G}^*$	ROC AUC	$\|\mathcal{G}^* - \mathcal{G}\|_F$	$\|\mathcal{G}^{\text{rand}} - \mathcal{G}\|_F$	N. edges($\mathcal{G}$)	N. edges($\mathcal{G}^*$)
Barabasi	0.989 ± 0.001	17 ± 2	26 ± 1	1360 ± 104	291
Erdos	0.94 ± 0.06	36.8 ± 0.8	44 ± 2	3131 ± 156	2092

Table 1: Results on ground-truth random graphs inference. $\mathcal{G}^*$ ($\mathcal{G}$) labels ground-truth (discovered) graph. $\|\cdot\|_F$ labels Frobenius norm. Error bars are computed from 10 random model initializations.

4 Inferring Ground-Truth Graphs

Before testing the behavior of our methodology on natural language data, we evaluate the ability of the model to infer hidden graph structures from sequential data in a controlled experiment. To this end, we define a synthetic language model with an underlying, ground-truth graph $\mathcal{G}^*$ as follows: Given a graph $\mathcal{G}^*$ with K nodes, and a vocabulary of random tokens $\mathcal{V}$ of size V, we assign one random bag of tokens (i.e. one pdf over $\mathcal{V}$) to each node of the graph. Let the K random bags be the set $\mathcal{E}$ of K symbols $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_K\}$ of the synthetic language model. We then sample N uniform random walks of length L over $\mathcal{G}^*$, and sample one random token from each symbol (i.e. from each random bag) along the walks. The result is a set of random token sequences of the same length as that of the random walks.

A simple proof-of-concept. We consider a problem in which the set of symbols $\mathcal{E}$ is known, so that the ground-truth graph $\mathcal{G}^*$ has a fixed labeling. This setting will allow for simple comparison between $\mathcal{G}^*$ and our inferred graphs. Following the procedure above we generate two datasets from two ground-truth graphs with different topologies. One sampled from the Barabási-Albert model (Barabási and Albert, 1999), the other from the Erdös-Rényi model (Erdös and Rényi, 1959). We set both graphs to have $K = 100$ symbols, and the token sequences to have length $L = 10$. Given these sets of random token sequences, the task is to infer the hidden, ground-truth graphs $\mathcal{G}^*$. To solve it we first replace BERT in Fig. 1 with a 2-block Transformer encoder (Vaswani et al., 2017) and note that, since the symbols are known, the likelihood of the model is simply given by $\prod_{i=1}^{L} \mathbf{e}_{j_i}$ where, as before, j_i denotes the index of the non-zero component of $\mathbf{z}_i$. We then train this simplified model by maximizing the HSN objective function. Further details about the graph model parameters, the dataset generation procedure and statistics, training procedure, and model sizes and hyperparameters can be found in Appendix E. The synthetic datasets are available in the source code too.

Results. Table 1 shows our results for our two synthetic datasets. Specifically, we compute the Area Under the Receiver Operating Characteristic Curve (ROC AUC) of our model $q_\phi(\mathbf{A})$ with respect to $\mathcal{G}^*$, and the Frobenius norm between q_ϕ and two graphs: the ground-truth one $\mathcal{G}^*$, and a second random graph $\mathcal{G}^{\text{rand}}$ sampled from the same random graph model as $\mathcal{G}^*$. We train ten (10) models in total and display the mean and standard deviation of our results. We also use a different $\mathcal{G}^{\text{rand}}$ for each calculation run. The first metric shows that $q_\phi(\mathbf{A})$ correctly predicts the edges of $\mathcal{G}^*$, whereas the other two metrics show that $\mathcal{G} \sim q_\phi(\mathbf{A})$ is closer to $\mathcal{G}^*$, than to any other random graph sampled from the same distribution. The last two columns in Table 1 show however that $q_\phi(\mathbf{A})$ tends to generate denser graphs as compared to the target. This overproduction of edges can be understood as being caused by the prior graph model, which is chosen to be an Erdös-Rényi model (see Section 3.1). The expected number of edges of this model is $\binom{K}{2} p$, where p is the prior edge probability. In our experiments we have $K = 100$, and choose $p = 0.2$ (0.6) when training the model on the Barabasi (Erdös) dataset (see Appendix E). The average number of edges of the prior graph model is therefore 990 (2970) for the Barabasi (Erdös) dataset. During training, the posterior model tries to fit the data, while simultaneously being close (in distribution) to the prior, which explains the overproduction of edges. Using different priors, like e.g. a sparse graph model, would reduce the average number of edges of the posterior graph distribution. See also Table 5 in Appendix E, where we report results from HSN with smaller prior edge probabilities.

5 Natural Language HSN

In the previous section we demonstrated that HSN can indeed infer hidden graph structure from sequential data in a simple setting. We now move on to our main problem: representation learning from natural language. We leverage pretrained BERT and GPT-2 language models as encoder

Model	PTB		YAHOO		YELP	
	PPL	MI	PPL	MI	PPL	MI
GPT2	24.23	–	22.00	–	23.40	–
iVAE$_{MI}$	53.44	12.50	47.93	10.70	36.88	11.00
Optimus$_A$	23.58	3.78	22.34	5.34	21.99	2.54
Optimus$_B$	35.53	8.18	29.92	9.18	24.59	9.13
HSN$_{(50, 5)}$	22.47	9.50	**20.99**	10.42	**19.72**	10.04
HSN$_{(50, 20)}$	30.38	**26.06**	22.84	**22.81**	21.60	**24.93**
HSN$_{(100, 5)}$	**20.25**	9.30	21.01	11.21	19.82	10.20
HSN$_{(100,20)}$	25.48	23.77	21.98	16.13	21.13	18.85

Table 2: Perplexity (PPL) (lower is better) and mutual information (MI). Optimus$_{A, B}$ label models with best PPL and MI, respectively (with $\lambda = 0.05, 1$) (Li et al., 2020) . GPT2 results are taken from Li et al. (2020). iVAE$_{MI}$ was taken from Fang et al. (2019). We sampled 100 (10) random walks (graphs) to estimate the PPL. End-of-sequence tokens are kept during evaluation.

Figure 2: Empirical degree distributions of inferred schema networks against that of an Erdös-Rényi graph with $p = 0.5$. Results correspond to HSN$_{(50, 5)}$. The graphs are sampled 500 times. Note that HSN differ from simple random graphs.

and decoder, respectively, which we finetune to discover schema networks encoding natural language datasets. We consider three widely used public datasets, namely the Penn Treebank (PTB) (Marcus et al., 1993), Yahoo and Yelp (Yang et al., 2017) corpora, and evaluate the quality of the inferred representations via the mutual information (MI) between schemata and data, and the perplexity (PPL) of the model. The latter is estimated through Monte Carlo samples. We compare HSN against a pretrained GPT-2, fine-tuned during a single epoch. We also compare against two VAE language models: iVAE$_{MI}$ (Fang et al., 2019) and Optimus (Li et al., 2020). The former implements both encoder and decoder as one-layer LSTMs (Hochreiter and Schmidhuber, 1997). The latter does it via pretrained BERT and GPT-2, just as HSN. Further details on the definition and computation of the evaluation metrics, as well as on hyperparameters and training procedures can be found in Appendix F. In what follows we explore HSNs with $K = \{50, 100\}$ symbols and hidden random walks of $L = \{5, 20\}$ steps. Let us label these configurations as HSN$_{(K,L)}$.

Results. As shown in Table 2, HSN achieves a much better performance than all baselines under both metrics, which implies that (i) HSN successfully encodes sentences into symbol sequences, and that (ii) our modified GPT-2 can effectively interpret symbolic representations. Longer random walks clearly have more capacity and, accordingly, encode more information. In contrast, shorter random walks perform better wrt. PPL, which illustrates the usual trade-off between language modeling and representation learning, typical of VAE (Li et al., 2020). We speculate the reason why the
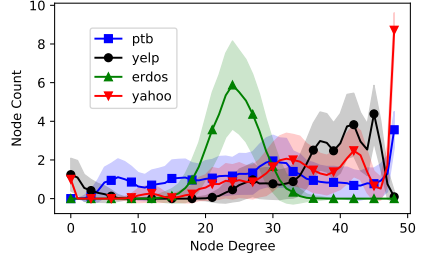
structured representations of HSN help with language modeling might be, that they allow for a sequential composition of the sentences' semantic components via reusable units. This feature might yield more efficient encodings than dense representations do. HSN also allows to infer (graph) connections not directly seen during training, that account for unseen symbol combinations, and hence (might account for) unseen sentences. We additionally report in Appendix F the averaged KL and likelihood values from five HSN, trained with different random initializations, against the baselines. Now we investigate the structure and semantics of the inferred schema networks.

Structure of hidden schema networks. We characterize the structure of the schema networks we inferred in terms of five statistics, namely their diameter, average distance, clustering coefficient, number of connected components and degree distribution. We define all these and report their computed values, for all HSN configurations and all datasets, in Appendix F. One observation we can immediately make about these results is that the schema networks discovered in each corpora tend to have smaller average distances, and much larger clustering coefficients, than any random graph of the same size, sampled from a maximum-entropy ($p = 0.5$) Erdös-Rényi model. Interestingly enough, the combination of these two features defines the so-called *small-world structure* (Watts and Strogatz, 1998). Now, intuitively speaking, a larger clustering coefficient implies that a random walker starting from a given node "k" will have a larger number of paths bringing it back to "k". In such an scenario, random walkers tend to cluster in neighborhoods around their starting point – a
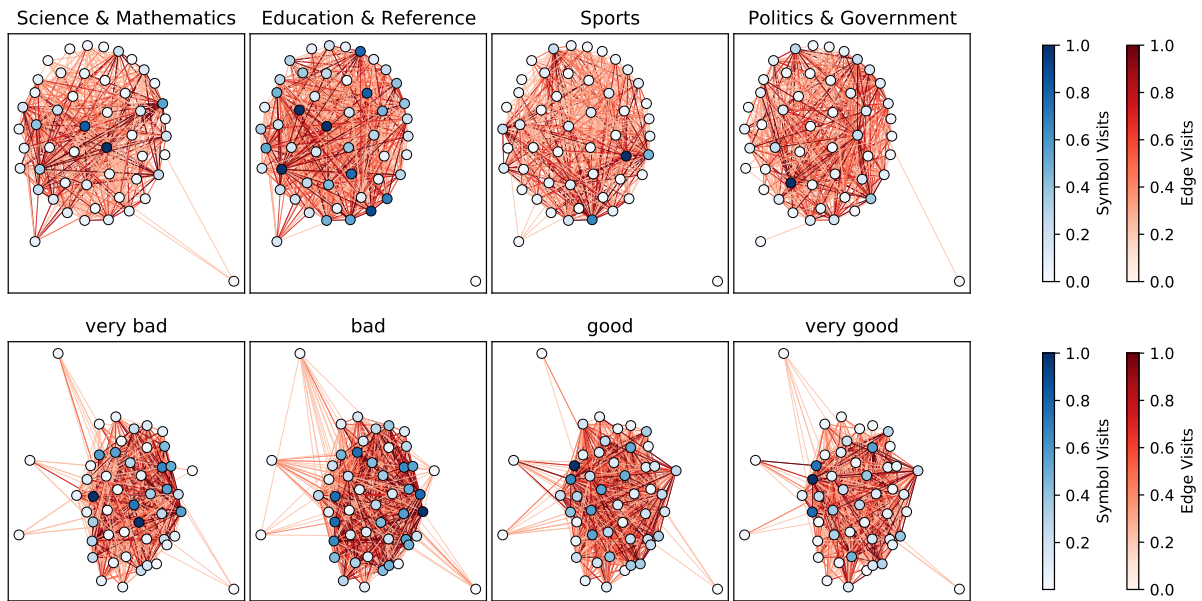
Figure 3: Schema distributions inferred from $\text{HSN}_{(50, 5)}$ for four subsets of the Yahoo (top) and Yelp (bottom) corpora. The node positions in the figure are consistent among labels and were computed using a force-directed embedding of the global graph $\mathcal{G}$.

property that could help encode different semantic aspects in different regions of $\mathcal{G}$. Similarly, one could also expect schemata composed of repeated symbols. We close this subsection with Figure 2, which displays the difference between the degree distributions of the schema networks and that of purely random graphs.

Schemata and semantics. Taking advantage of the labels available to both Yahoo and Yelp, we display in Figure 3 the schema distributions over the schema networks for 4 subsets of these datasets. Similar plots for all subsets of both corpora, extracted with all HSN configurations, can be found in Appendix F. Note how the "hot" symbols per category reside on different regions of the graphs – as suspected already from the large clustering coefficient of $\mathcal{G}$ – and yet, the "Science & Math" schemata (both nodes and edges) of Yahoo are *closer* to the "Education & Reference" schemata than to the "Sport" schemata, where closer nodes in the figure indicate well-connected nodes in the underlying graph $\mathcal{G}$. These findings (qualitatively) indicate that the schemata indeed encode semantic notions of their corpora. A similar picture holds for Yelp. We also introduce and explore "schema interpolations" in Appendix F.3, and observe that the translation procedure we employ successfully interpolates between categories of both Yelp and Yahoo. These experiments hint too at the different semantic notions encoded by HSN.

To get a more quantitative evaluation of the semantics encoded in the schema networks, one can also investigate how the distribution of attention weights, in the HSN decoder, and wrt. the symbols changes, as the decoder processes specific words. We exemplify this idea in Appendix G.

6 Commonsense Reasoning Generation

In the previous section we empirically showed that HSN is able to infer schema networks from natural language datasets, whose symbols encode different notions of semantics. Next, we explore training "reasoning" models on the schema networks inferred from commonsense knowledge databases, and using these secondary models to sample novel (reasoning) paths and enhance the performance of LPLM on commonsense *If-Then* reasoning tasks. Let us start by defining this task.

It has been recently proposed that LPLMs fine-tuned on (natural language) knowledge graph (KG) tuples, can express their encoded knowledge through language generation, thereby providing commonsense knowledge on demand (Bosselut et al., 2019; Hwang et al., 2020). Consider a KG of natural language tuples of the form $(\mathbf{s}, \mathbf{r}, \mathbf{o})$, where $\mathbf{s} = (\mathbf{x}_1^s, \dots, \mathbf{x}_{|\mathbf{s}|}^s)$ labels the *phrase subject* of the tuple, $\mathbf{r} = \mathbf{x}^r$ is the *relation token* and $\mathbf{o} = (\mathbf{x}_1^o, \dots, \mathbf{x}_{|\mathbf{o}|}^o)$ is the *phrase object* of the tuple. In this setting, commonsense reasoning consists in

	HSN$_{(50, 20)}$	HSN$^{AR}_{(50, 20)}$	COMET$_{(GPT2)}$	COMET$_{(GPT2-XL)}$	COMET$_{(BART)}$
BLEU-2	**0.462**	0.129	0.225	0.300	0.330
BERT Score	**0.694**	0.374	0.385	0.638	0.650

Table 3: Object generation quality. COMET$_{(GPT2-XL)}$ and COMET$_{(BART)}$ results were extracted from (Hwang et al., 2020). COMET$_{(GPT2)}$ was computed by us. All models use greedy decoding for *all* text prefixes in the dataset.

generating novel objects $\mathbf{o}$, given $\mathbf{s}$ and $\mathbf{r}$.

The COMET framework of Bosselut et al. (2019) finetunes GPT-like models by maximizing the likelihood of the object, conditioned on the sequence $(\mathbf{s}, \mathbf{r}) = (\mathbf{x}_1^s, \dots, \mathbf{x}_{|\mathbf{s}|}^s, \mathbf{x}^r)$. We instead approach the problem of commonsense knowledge generation with a three-step process: (I) we auto-encode the KG tuples using HSN so that the first $\frac{L}{2}$ symbols of the schemata encode $(\mathbf{s}, \mathbf{r})$ only, while their last $\frac{L}{2}$ symbols encode the complete $(\mathbf{s}, \mathbf{r}, \mathbf{o})$ tuple; (II) we train an autoregressive (AR) "reasoning" model, that takes $(\mathbf{s}, \mathbf{r})$ and $\mathbf{z}_{1:\frac{L}{2}}$ (the first $\frac{L}{2}$ schema symbols inferred in step I) as inputs, and learns to generate the second half of the schema $\mathbf{z}_{\frac{L}{2}+1:L}$ (the half that encodes the object); (III) given unseen $(\mathbf{s}, \mathbf{r})$ pairs and their associated first half-schemata $\mathbf{z}_{1:\frac{L}{2}}$, we sample novel, second half-schemata $\mathbf{z}_{\frac{L}{2}+1:L}$ from the trained AR model, and use them to condition the HSN decoder and generate novel objects.

To do step I we write the random walk posterior distribution $q_\phi(\mathbf{z}_{1:L}|\mathbf{s}, \mathbf{r}, \mathbf{o})$ of HSN in the product form $q_\phi(\mathbf{z}_{1:\frac{L}{2}}|\mathbf{s}, \mathbf{r})q_\phi(\mathbf{z}_{\frac{L}{2}+1:L}|\mathbf{z}_{\frac{L}{2}}, \mathbf{s}, \mathbf{r}, \mathbf{o})$, where we omitted the explicit dependence on $\mathbf{A}$, in all terms, for clarity. Both terms in the product are modeled with the same HSN encoder architecture as before (see Fig. 1), but share a single, pretrained BERT model. To do step II we introduce an AR "reasoning" model $p_\theta(\mathbf{z}_{\frac{L}{2}+1:L}|\mathbf{z}_{1:\frac{L}{2}}, \mathbf{s}, \mathbf{r})$, that processes the first half-schema $\mathbf{z}_{1:\frac{L}{2}}$ via a recurrent neural net, and the pair $(\mathbf{s}, \mathbf{r})$ via a second, pretrained BERT model. This AR model is trained to generate the second half-schema on samples from $q_\phi(\mathbf{z}_{\frac{L}{2}+1:L}|\mathbf{z}_{\frac{L}{2}}, \mathbf{s}, \mathbf{r}, \mathbf{o})$. Finally, we do step III by concatenating the first half-schema, sampled from the posterior model, with the second half-schema, sampled from the AR model. The novel reasoning path is used to condition the HSN decoder to generate novel objects. See Appendix H for additional details.

For this preliminary study we focus on the ATOMIC dataset (Sap et al., 2019), evaluate the quality of the generated objects with both, BLEU-2 (Papineni et al., 2002) and BERT Score (Zhang* et al., 2020) metrics, and compare against GPT-2, GPT-2-XL and BART, all trained within the COMET framework (Hwang et al., 2020). We also fix the number of symbols $K = 100$, and the random walk length $L = 20$.

Results. We find that our best AR models only learn about 58% of the second half-schema. That is, about half of the symbols encoding the complete $(\mathbf{s}, \mathbf{r}, \mathbf{o})$ tuple. Table 3 shows the scores we obtain when conditioning the decoder on the schemata sampled from the posterior (which sees the target object), and from the AR model (which does not sees the object): HSN$_{(50, 20)}$ and HSN$^{AR}_{(50, 20)}$, respectively. That HSN$_{(50, 20)}$ outperforms all baselines is expected, for HSN can successfully encode its inputs, as we empirically showed in Section 5 above. We find interesting that HSN$^{AR}_{(50, 20)}$ is comparable to COMET$_{(GPT2)}$ in BERT Score, despite leveraging an imperfect AR model, which only reproduces about 58% of the correct encoding. The scores achieved by HSN$_{(50, 20)}$ imply that improving upon the AR model should increase the performance of HSN$^{AR}_{(50, 20)}$, even beyond that of COMET$_{(BART)}$. We speculate that the AR "reasoning" model struggles to infer the schema distribution because of the complexity of the latter. This complexity is direct consequence of the object-encoding scheme we use (step I above), for which e.g. missing one single symbol can lead to random walks completely off. We shall investigate different encoding schemes in future work.

7 Conclusion

We have introduced a novel representation learning algorithm for natural language that (i) enforces, via inductive biases, relational structures which allow for compositionality onto the output representations of LPLM, and (ii) allows for symbolic reasoning via random walk processes on the discovered graphs. Future work shall investigate replacing both encoding and decoding models with more expressive LPLM, and explore training different "reasoning" models and encoding schemes, on natural language understanding tasks.

Limitations

Although our methodology is agnostic to the specific sequence-processing models used as encoder and decoder, its main purpose is to be used together with LPLM. This clearly limits its use to institutions and users with access to computing facilities able to handle such models. Indeed, although the encoder of HSN can successfully be trained while keeping the weights of BERT (or any other LPLM) frozen, a feature which reduces training time, the GPT-based decoder does need to be fine-tuned to learn how to interpret the inferred symbols. This last points further limits the use of our methodology to "not-so-large" decoder models. That being said, fine-tuning HSN decoders via LoRA (Hu et al., 2022a) is an exciting option that can possibly solve (or help with) this last issue. We shall explore using LoRA with HSN in the near future.

Ethics Statement

We do not find any ethical concerns either with the model or the datasets we used (which are publicly available).

Acknowledgements

References

Joshua T Abbott, Joseph L Austerweil, and Thomas L Griffiths. 2012. Human memory search as a random walk in a semantic network. In *Annual Conference on Neural Information Processing Systems*, pages 3050–3058.

Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science*, 286(5439):509–512.

Frederic C Bartlett. 1932. *Remembering: A Study in Experimental and Social Psychology*. Cambridge University Press.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3:1137–1155.

Hanoz Bhathena, Angelica Willis, and Nathan Dass. 2020. Evaluating compositionality of sentence representation models. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 185–193. Association for Computational Linguistics.

Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

Ned Block. 1986. Advertisement for a semantics for psychology. *Midwest studies in philosophy*, 10:615–678.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 4762–4779. Association for Computational Linguistics.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Ronald J Brachman. 1977. What's in a concept: structural foundations for semantic networks. *International journal of man-machine studies*, 9(2):127–152.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

America Chambers, Padhraic Smyth, and Mark Steyvers. 2010. Learning concept graphs from text with stick-breaking priors. In *Annual Conference on Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

Xiaojun Chen, Shengbin Jia, and Yang Xiang. 2020. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948.

Allan M Collins and M Ross Quillian. 1969. Retrieval time from semantic memory. *Journal of verbal learning and verbal behavior*, 8(2):240–247.

Thomas M Cover and Joy A Thomas. 1991. Information theory and statistics. *Elements of information theory*, 1(1):279–335.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

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*.

Jesse Dunietz, Greg Burnham, Akash Bharadwaj, Owen Rambow, Jennifer Chu-Carroll, and Dave Ferrucci. 2020. To test machine comprehension, start by defining comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7839–7859. Association for Computational Linguistics.

Paul Erdös and Alfréd Rényi. 1959. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290–297.

Le Fang, Chunyuan Li, Jianfeng Gao, Wen Dong, and Changyou Chen. 2019. Implicit deep latent variable models for text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3946–3956. Association for Computational Linguistics.

JR Firth. 1957. A synopsis of linguistic theory, 1930—55. *Studies in Linguistic Analysis. London: Black*.

Jerry A Fodor and Ernest Lepore. 2002. *The compositionality papers*. Oxford University Press.

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. 2019. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics.

Jesús Gómez-Gardenes and Vito Latora. 2008. Entropy rate of diffusion processes on complex networks. *Physical Review E*, 78(6):065102.

Thomas L Griffiths, Mark Steyvers, and Alana Firl. 2007a. Google and the mind: Predicting fluency with pagerank. *Psychological science*, 18(12):1069–1076.

Thomas L Griffiths, Mark Steyvers, and Joshua B Tenenbaum. 2007b. Topics in semantic representation. *Psychological review*, 114(2):211.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022a. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. 2017. Learning discrete representations via information maximizing self-augmented training. In *Proceedings of the 34th International Conference on Machine Learning*, page 1558–1567. JMLR.org.

Ziniu Hu, Yichong Xu, Wenhao Yu, Shuohang Wang, Ziyi Yang, Chenguang Zhu, Kai-Wei Chang, and Yizhou Sun. 2022b. Empowering language models with knowledge graph reasoning for open-domain question answering. In *EMNLP*.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: how do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Jena D Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2020. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. *arXiv preprint arXiv:2010.05953*.

Ray Jackendoff. 1978. An argument about the composition of conceptual structure. *American Journal of Computational Linguistics*, pages 69–73.

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

Łukasz Kaiser and Samy Bengio. 2018. Discrete autoencoders for sequence models. *arXiv preprint arXiv:1801.09797*.

Łukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2390–2399. PMLR.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G P Shrivatsa Bhargav, and Mo Yu. 2021. Leveraging Abstract Meaning Representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3884–3894. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

Renaud Lambiotte, Roberta Sinatra, J-C Delvenne, Tim S Evans, Mauricio Barahona, and Vito Latora. 2011. Flow graphs: Interweaving dynamics and structure. *Physical Review E*, 84(1):017102.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics.

Bohan Li, Junxian He, Graham Neubig, Taylor Berg-Kirkpatrick, and Yiming Yang. 2019. A surprisingly effective fix for deep latent variable modeling of text. *arXiv preprint arXiv:1909.00868*.

Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 2020. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4678–4699. Association for Computational Linguistics.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094. Association for Computational Linguistics.

Kaixin Ma, Jonathan Francis, Quanyang Lu, Eric Nyberg, and Alessandro Oltramari. 2019. Towards generalizable neuro-symbolic systems for commonsense question answering. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 22–32. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.

Eric Ed Margolis and Stephen Ed Laurence. 1999. *Concepts: Core Readings.* The MIT Press.

James L McClelland and Timothy T Rogers. 2003. The parallel distributed processing approach to semantic cognition. *Nature reviews neuroscience*, 4(4):310–322.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Marvin Minsky. 1974. A framework for representing knowledge. *MIT-AI Laboratory Memo 306*.

Marvin Minsky. 1975. Minsky's frame system theory. In *TINLAP*, volume 75, pages 104–116.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *proceedings of ACL-08: HLT*, pages 236–244.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Juyong Park and Mark EJ Newman. 2004. Statistical mechanics of networks. *Physical Review E*, 70(6):066117.

Jean Piaget. 1948. *Le langage et la pensée chez l'enfant: Études sur la logique de l'enfant.* Delachaud & Niestlé.

M Ross Quillan. 1966. Semantic memory. Technical report, Bolt Beranek and Newman Inc., Cambridge MA.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. *Preprint.*

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. 2021. Probing the probing paradigm: Does probing accuracy entail task relevance? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3363–3377, Online. Association for Computational Linguistics.

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.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

David E Rumelhart. 2017. Schemata: The building blocks of cognition. In *Theoretical issues in reading comprehension*, pages 33–58. Routledge.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

Ray Solomonoff and Anatol Rapoport. 1951. Connectivity of random nets. *The bulletin of mathematical biophysics*, 13(2):107–117.

Karl Stratos and Sam Wiseman. 2020. Learning discrete structured representations by adversarially maximizing mutual information. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9144–9154. PMLR.

Alon Talmor, Ori Yoran, Ronan Le Bras, Chandra Bhagavatula, Yoav Goldberg, Yejin Choi, and Jonathan Berant. 2021. CommonsenseQA 2.0: Exposing the limits of AI through gamification. In *Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. 2011. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285.

Paul Thagard. 1984. Frames, knowledge, and inference. *Synthese*, 61(2):233–259.

Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. 2017. Neural discrete representation learning. In *Annual Conference on Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Annual Conference on Neural Information Processing Systems*, pages 5998–6008.

Pat Verga, Haitian Sun, Livio Baldini Soares, and William Cohen. 2021. Adaptable and interpretable neural MemoryOver symbolic knowledge. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3678–3691. Association for Computational Linguistics.

Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440–442.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Peter West, Chandra Bhagavatula, Jack Hessel, Jena D Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2021. Symbolic knowledge distillation: from general language models to commonsense models. *arXiv preprint arXiv:2110.07178*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3881–3890. PMLR.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546. Association for Computational Linguistics.

Lang Yu and Allyson Ettinger. 2020. Assessing phrasal representation and composition in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4896–4907. Association for Computational Linguistics.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2019. Infovae: Balancing learning and inference in variational autoencoders. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5885–5892.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1098–1107. Association for Computational Linguistics.

Zachary M Ziegler, Luke Melas-Kyriazi, Sebastian Gehrmann, and Alexander M Rush. 2019. Encoder-agnostic adaptation for conditional language generation. *arXiv preprint arXiv:1908.06938*.

Appendix

A Pseudo-Self Attention Mechanism Revisited

The attention mechanism of the original Transformers (Vaswani et al., 2017) is defined as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) =$$
$$\text{softmax}\left(D^{-\frac{1}{2}}\,\mathbf{Q}\cdot\mathbf{K}^T\right)\cdot\mathbf{V}, \quad (7)$$

where $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V} \in \mathbb{R}^{T\times D}$ are sets of queries, keys and values, respectively, given by a sequence of T, D-dimensional vectors, packed into matrices. In practice, these queries, keys and values are projected many times with different learnable, linear maps. The $\text{Attention}(\cdot)$ operation (Eq. 7) is performed on these different projections in parallel, whose outputs are then concatenated and projected once more with a final, linear map. The complete operation is known as Multi-head Attention (Vaswani et al., 2017), and we use this notation in Fig. 1 of the main text.

Now, the question is how to condition GPT-2 on the schema $\mathbf{e}_{j_1:j_L}$. Given a sequence of input representations $\mathbf{u}_{1:T}$, the *self*-attention mechanism in GPT-2 is obtained by choosing $\mathbf{Q} = \mathbf{u}_{1:T}\cdot\mathbf{W}_Q, \mathbf{K} = \mathbf{u}_{1:T}\cdot\mathbf{W}_K$ and $\mathbf{V} = \mathbf{u}_{1:T}\cdot\mathbf{W}_V$, all in $\mathbb{R}^{T\times D}$, with $\mathbf{W}_Q, \mathbf{W}_K$ and $\mathbf{W}_V \in \mathbb{R}^{D\times D}$ pretrained matrices. We leverage a pseudo-self attention (PSA) mechanism (Ziegler et al., 2019) that augments the key and value matrices in their first L rows, with projections of $\mathbf{e}_{j_1:j_L}$ so that

$$\tilde{\mathbf{K}} = \begin{pmatrix} \mathbf{e}_{j_1:j_L}\cdot\mathbf{W}_K^e + \mathbf{p}_{\text{enc}} \\ \mathbf{K} \end{pmatrix},$$

$$\tilde{\mathbf{V}} = \begin{pmatrix} \mathbf{e}_{j_1:j_L}\cdot\mathbf{W}_V^e + \mathbf{p}_{\text{enc}} \\ \mathbf{V} \end{pmatrix}, \quad (8)$$

both in $\mathbb{R}^{(L+T)\times D}$, where $\mathbf{p}_{\text{enc}}$ is a positional encoding, just as the one used in the original Transformer implementation (Vaswani et al., 2017). The latter informs GPT-2 about the ordering of the symbols in the schema, as selected by the random walk process. PSA is then simply given by Eq. 7 with the keys and values replaced with the augmented ones, $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{V}}$. The $\mathbf{W}_K^e, \mathbf{W}_V^e$ here are randomly initialized, learnable parameters mapping the schemata onto the decoder self-attention, D-dimensional space, and we have as many of them as layers in GPT-2. Therefore this mechanism allows GPT-2 to attend to the projected schema at each of its layers, with a minimal addition of untrained parameters (Ziegler et al., 2019).

B Hidden Schema Networks Algorithm

Algorithm 1: HSN Training (ϕ, ψ)

---

foreach *minibatch* $\mathbf{x}_{1:T} \sim p(\mathcal{D})$ **do**

(1) Sample schema network from posterior graph model:

$$\mathbf{A} \quad \sim \quad q_\phi(\mathbf{A}),$$

(2) Compute parameters of posterior random walk model:

$$\mathbf{h}_1^{\text{enc}}, \mathbf{h}_2^{\text{enc}}\ldots, \mathbf{h}_L^{\text{enc}} \;=\; \mathbf{h}_\phi^{\text{enc}}(\mathbf{x}_{1:T}),$$
$$\boldsymbol{\rho}(\phi) \;=\; \text{softmax}(\mathbf{h}_1^{\text{enc}}),$$
$$Q_{k,j}^{[i]}(\phi) \;=\; \frac{f_k^{[i]}(\phi)\,A_{kj}}{\sum_m f_m^{[i]}(\phi)\,A_{mj}},$$

with $\mathbf{f}^{[1]}, \ldots, \mathbf{f}^{[L-1]} = \exp(\mathbf{h}_{2:L}^{\text{enc}})$

(3) Compute parameters of prior random walk model:

$$P_{k,j} = \frac{f_k\,A_{kj}}{\sum_{i=1}^K f_i\,A_{ij}}$$

(4) Sample random walks from posterior distribution:

$$\mathbf{z}_{1:L} \sim q_\phi(\mathbf{z}_{1:L}|\mathbf{x}_{1:T}, \mathbf{A})$$

(5) Decode sentence:

for $i = 0$ **to** $T - 1$ **do**
$\quad \mathbf{x}_i \sim p_\theta(\mathbf{x}_i|\mathbf{x}_{<i}, \mathbf{e}_{j_1:j_L}),$
$\quad \boldsymbol{\pi}_i =$
$\quad\quad \text{softmax}(\mathbf{W}\cdot\mathbf{h}_\theta^{\text{dec}}(\mathbf{x}_{<i}, \mathbf{e}_{j_1:j_L}))$
end

(6) Compute loss and back-propagate:

$$\mathcal{L}[\theta, \phi] = \frac{1}{N}\sum_{n=1}^N \mathbb{E}_{q_\phi(\mathbf{z}_{1:L}|\mathbf{x}_{1:T}^{(n)}, \mathbf{A})q_\phi(\mathbf{A})}$$
$$\left\{\log p_\theta(\mathbf{x}_{1:T}^{(n)}|\mathbf{z}_{1:L})\right\}$$
$$- \mathbb{E}_{q_\phi(\mathbf{A})}\text{KL}\left[q_\phi^*(\mathbf{z}_{1:L}|\mathbf{A}); p(\mathbf{z}_{1:L}|\mathbf{A})\right]$$
$$- \text{KL}[q_\phi(\mathbf{A}); p(\mathbf{A})]$$

end

C Inference model: Full Equations

C.1 Posterior over (global) graph

We model the posterior over the graph assigning Bernoulli variables to its links, but we let the probability of observing each link depend on the global symbols

$$q_\phi(\mathbf{A}) = \prod_{i,j} p_\phi(\mathbf{e}_i, \mathbf{e}_j)^{a_{ij}} \left(1 - p_\phi(\mathbf{e}_i, \mathbf{e}_j)\right)^{1-a_{ij}}$$

$$\text{where } p_\phi(\mathbf{e}_i, \mathbf{e}_j) = \text{sigmoid}(g_\phi(\mathbf{e}_i, \mathbf{e}_j)),$$

with $g_\phi : \mathcal{E} \times \mathcal{E} \to \mathbb{R}$ a deep neural network, and $p_\phi(\mathbf{e}_i, \mathbf{e}_j) \in [0, 1]$, for all $\mathbf{e}_i \in \mathcal{E}$, the link probabilities. Our reasoning here is that the network g_ϕ should infer graphs connecting symbols which are semantically related via the encoded sentences.

C.2 Posterior over random walks (encoder model)

We model the posterior probability over random walks on $\mathcal{G}$ as

$$q_\phi(\mathbf{z}_{1:L}|\mathbf{x}_{1:T}, \mathbf{A}) = \left(\prod_{i=1}^{K} \rho_i(\mathbf{x}_{1:T}, \phi)^{z_1^i}\right)$$

$$\cdot \prod_{i=2}^{L} \left(\prod_{j=1}^{K}\prod_{k=1}^{K} \left(Q_{k,j}^{[i-1]}(\mathbf{x}_{1:T}, \mathbf{A}, \phi)\right)^{z_i^k z_{i-1}^j}\right),$$
(9)

where instead of having a single transition probability matrix, we have a sequence of them, thereby allowing the posterior to capture inhomogeneous random walks. Note that we could have also chosen a mean-field decomposition along the steps of the random walk, simply by either ignoring the dependency on the graph, or making the graph fully connected (see Appendix D.5). Going back to Eq. 9, we model the probabilities over the starting point of the random walks and the transition matrices as follows

$$\boldsymbol{\rho}(\mathbf{x}_{1:T}, \phi) = \text{softmax}(\mathbf{h}_1^{\text{enc}}),$$

$$Q_{k,j}^{[i]}(\mathbf{x}_{1:T}, \mathbf{A}, \phi) = \frac{f_k^{[i]}(\mathbf{x}_{1:T}, \phi)\, A_{kj}}{\sum_m f_m^{[i]}(\mathbf{x}_{1:T}, \phi)\, A_{mj}},$$

$$\text{with } \mathbf{f}^{[1]}, \dots, \mathbf{f}^{[L-1]} = \exp(\mathbf{h}_{2:L}^{\text{enc}}),$$

where $\mathbf{h}_1^{\text{enc}}, \mathbf{h}_2^{\text{enc}}, \dots, \mathbf{h}_L^{\text{enc}} \in \mathbb{R}^D$ is the sequence of outputs of a deep neural network model $\mathbf{h}_\phi^{\text{enc}}(\mathbf{x}_{1:T})$ processing the input sequence of T words. The model $\mathbf{h}_\phi^{\text{enc}}(\mathbf{x}_{1:T})$ must then map a sequence of T

vectors to a sequence of L vectors. We define $\mathbf{h}_\phi^{\text{enc}}$ by a *pretrained* BERT model (Devlin et al., 2018), followed by a single Transformer block, randomly initialized. The Transformer block processes the T (D-dimensional) outputs from BERT as keys and values, together with a set of L learnable vectors $\mathbf{q}_{1:L}$ as queries. The right hand side of Figure 1 illustrates the complete encoder architecture.

D On the HSN Training Objective

Following standard methods (Bishop, 2006) one readily can show that the Evidence Lower Bound (ELBO) of the Hidden Schema Network model is given by

$$\mathcal{L}[\theta, \phi] = \frac{1}{N} \sum_{n=1}^{N} \Big\{$$

$$\mathbb{E}_{q_\phi(\mathbf{z}_{1:L}|\mathbf{x}_{1:T}^{(n)}, \mathbf{A})q_\phi(\mathbf{A})} \left[\log p_\theta(\mathbf{x}_{1:T}^{(n)}|\mathbf{z}_{1:L})\right]$$

$$- \mathbb{E}_{q_\phi(\mathbf{A})} \text{KL}\left[q_\phi(\mathbf{z}_{1:L}|\mathbf{x}_{1:T}^{(n)}, \mathbf{A}); p(\mathbf{z}_{1:L}|\mathbf{A})\right]\Big\}$$

$$- \text{KL}[q_\phi(\mathbf{A}); p(\mathbf{A})], \quad (10)$$

where $\text{KL}[\cdot]$ denotes the Kullback-Leibler (KL) divergence.

Note that this is *not* the training objective of the main text. There we maximize the ELBO together with the mutual information between sentences and schemata. We give details about this modified objective in subsections D.3 and D.4 below. Before getting into that, let us first calculate the explicit expressions for the two divergences above.

D.1 Kullback-Leibler Divergence Between Random Walks

For notational convenience we will not write the explicit dependence on the graph $\mathbf{A}$ in what follows. Using the explicit product form of the probabilities over walks leads to

$$\text{KL}[q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}^{(n)}); p(\mathbf{z}_{1:T})] =$$

$$\sum_{i=2}^{L} \mathbb{E}_{\hat{q}_\phi(\mathbf{z}_{i-1}|\mathbf{x}_{1:T}^{(n)})q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1}\mathbf{x}_{1:T}^{(n)})}$$

$$\left\{\log \frac{q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1}, \mathbf{x}_{1:T}^{(n)})}{p(\mathbf{z}_i|\mathbf{z}_{i-1})}\right\} + \text{KL}[q_\phi(\mathbf{z}_1); p(\mathbf{z}_1)],$$
(11)

where $\hat{q}_\phi(\mathbf{z}_i|\mathbf{x}_{1:T}^{(n)})$ is the aggregated probability over all walks until step i. Since the random walks

are Markovian, $\hat{q}$ can be explicitly written as

$$\hat{q}_\phi(\mathbf{z}_i|\mathbf{x}_{1:T}^{(n)}) = \prod_{1 \leq j < i} \mathbf{Q}^{[j]}(\mathbf{x}_{1:T}^{(n)}, \phi) \cdot \boldsymbol{\rho}(\mathbf{x}_{1:T}^{(n)}, \phi), \quad (12)$$

where the (posterior) transition matrices $\mathbf{Q}^{[i]}$ are defined in Eq. 6 of the main text. Using the generic random walk definition in Eqs. 3 we can readily write the argument of the expectation value in Eq. 11 above as

$$\log \frac{q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1}, \mathbf{x}_{1:T}^{(n)})}{p(\mathbf{z}_i|\mathbf{z}_{i-1})} = \sum_{k,j} z_i^k z_{i-1}^j \log \frac{Q_{k,j}^{[i-1]}(\mathbf{x}_{1:T}^{(n)}, \phi)}{P_{k,j}}, \quad (13)$$

which means we only need to compute the expectation of the product $z_i^k z_{i-1}^j$. This one can easily be shown to be

$$\mathbb{E}_{\hat{q}_\phi(\mathbf{z}_{i-1}|\mathbf{x}_{1:T}^{(n)})q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1}\mathbf{x}_{1:T}^{(n)})}\left[z_i^k z_{i-1}^j\right] = Q_{k,j}^{[i-1]}(\mathbf{x}_{1:T}^{(n)}, \phi)\, \hat{\rho}_j^{[i-1]}(\mathbf{x}_{1:T}^{(n)}, \phi), \quad (14)$$

where $\hat{\rho}_j^{[i]}(\mathbf{x}_{1:T}^{(n)}, \phi)$ is the jth class probability of $\hat{q}_\phi(\mathbf{z}_i|\mathbf{x}_{1:T}^{(n)})$, defined in Eq. 12.

Finally, the second KL term in Eq. 11 can be directly evaluated

$$\mathrm{KL}[q_\phi(\mathbf{z}_1); p(\mathbf{z}_1)] = \sum_{j=1}^{K} \rho_j(\mathbf{x}_{1:T}^{(n)}, \phi) \log \frac{\rho_j(\mathbf{x}_{1:T}^{(n)}, \phi)}{\rho_j}, \quad (15)$$

where $\rho_j(\mathbf{x}_{1:T}^{(n)}, \phi)$ and ρ_j are, respectively, the posterior and prior class probabilities for the random walks' starting points.

Putting all together we write

$$\mathrm{KL}[q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}^{(n)}); p(\mathbf{z}_{1:T})] =$$
$$\sum_{i=2}^{L} \sum_{k,j=1}^{K} Q_{k,j}^{[i-1]}(\mathbf{x}_{1:T}^{(n)}, \phi)\, \hat{\rho}_j^{[i-1]}(\mathbf{x}_{1:T}^{(n)}, \phi)$$
$$\times \log \frac{Q_{k,j}^{[i-1]}(\mathbf{x}_{1:T}^{(n)}, \phi)}{P_{k,j}}$$
$$+ \sum_{j=1}^{K} \rho_j(\mathbf{x}_{1:T}^{(n)}, \phi) \log \frac{\rho_j(\mathbf{x}_{1:T}^{(n)}, \phi)}{\rho_j} \quad (16)$$

D.2 Kullback-Leibler Divergence Between Random Graph Models

Since both prior and posterior graph models treat each edge in $\mathcal{G}$ as a Bernoulli random variable, we can write directly

$$\mathrm{KL}[q(\mathbf{A}); p(\mathbf{A})] =$$
$$\sum_{ij} \left\{ p_\phi(\mathbf{e}_i, \mathbf{e}_j) \log \left(\frac{p_\phi(\mathbf{e}_i, \mathbf{e}_j)}{p} \right) \right.$$
$$\left. + (1 - p_\phi(\mathbf{e}_i, \mathbf{e}_j)) \log \left(\frac{1 - p_\phi(\mathbf{e}_i, \mathbf{e}_j)}{1 - p} \right) \right\}, \quad (17)$$

where $p_\phi(\mathbf{e}_i, \mathbf{e}_j)$ is the posterior link probability, which is conditioned on the symbols connected by the link, and p is the global prior probability over all links.

D.3 Maximizing Mutual Information

We would like to maximize the mutual information between the word sequences in our dataset and the schema representations. We have argued that the training objective in the main text already includes such a mutual information term. To see this is indeed the case we need to workout some identities.

Let us, for simplicity of notation, consider two discrete variables $\mathbf{z}$ and $\mathbf{x}$, the last of which follows an unknown distribution $p_\mathcal{D}(\mathbf{x})$. What follow are identities

$$-\mathbb{E}_{p_\mathcal{D}(\mathbf{x})}\mathrm{KL}[q(\mathbf{z}|\mathbf{x}); p(\mathbf{z})]$$
$$= \mathbb{E}_{p_\mathcal{D}(\mathbf{x})}\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left\{ \log p(\mathbf{z}) - \log(\mathbf{z}|\mathbf{x}) \right\}$$
$$= H_q(\mathbf{z}|\mathbf{x}) + \sum_{\mathbf{x}} p_\mathcal{D}(\mathbf{x}) \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x})\left\{ \log p(\mathbf{z}) \right.$$
$$\left. + \log q^*(\mathbf{z}) - \log q^*(\mathbf{z}) \right\}$$
$$= H_q(\mathbf{z}|\mathbf{x}) - H_{q^*}(\mathbf{z})$$
$$+ \sum_{\mathbf{z}} q^*(\mathbf{z})\left\{ \log p(\mathbf{z}) - \log q^*(\mathbf{z}) \right\}$$
$$= -I(\mathbf{z}; \mathbf{x}) - \mathrm{KL}[q^*(\mathbf{z}); p(\mathbf{z})], \quad (18)$$

where

$$H_q(\mathbf{z}|\mathbf{x}) = -\sum_{\mathbf{x}} p_\mathcal{D}(\mathbf{x}) \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log q(\mathbf{z}|\mathbf{x}), \quad (19)$$

is the conditional entropy with respect to distribution q (see e.g. page 17 in Cover and Thomas (1991)) and

$$H_{q^*}(\mathbf{z}) = -\sum_{\mathbf{z}} q^*(\mathbf{z}) \log q^*(\mathbf{z}), \qquad (20)$$

is the entropy of distribution $q^*(\mathbf{z})$, which we define as the marginal (data-aggregated) distribution

$$q^*(\mathbf{z}) = \sum_{\mathbf{x}} p_{\mathcal{D}}(\mathbf{x}) q(\mathbf{z}|\mathbf{x}). \qquad (21)$$

Finally, we used the definition of mutual information

$$I(\mathbf{x}; \mathbf{z}) = H_{q^*}(\mathbf{z}) - H_q(\mathbf{z}|\mathbf{x}). \qquad (22)$$

See e.g. page 20 in Cover and Thomas (1991).

It follows from Eq. 18 that maximizing the ELBO (Eq. 10), together with the mutual information between word sequences and schemata, simply amounts to replacing the KL between the approximate posterior and prior random walk distributions, with the KL between the *aggregated posterior* and prior random walk distributions. To wit

$$I(\mathbf{z}_{1:L}; \mathbf{x}_{1:T}|\mathbf{A}) - \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{q_\phi(\mathbf{A})} \Big\{$$
$$KL\Big[q_\phi(\mathbf{z}_{1:L}|\mathbf{x}_{1:T}^{(n)}, \mathbf{A}); p(\mathbf{z}_{1:T}|\mathbf{A}) \Big] \Big\}$$
$$= -\mathbb{E}_{q_\phi(\mathbf{A})} KL\Big[q_\phi^*(\mathbf{z}_{1:L}|\mathbf{A}); p(\mathbf{z}_{1:T}|\mathbf{A}) \Big], \quad (23)$$

where we introduced the aggregated posterior over random walks wrt the word sequence

$$q_\phi^*(\mathbf{z}_{1:L}|\mathbf{A}) = \mathbb{E}_{p(\mathbf{x}_{1:T})} \Big[q_\phi(\mathbf{z}_{1:L}|\mathbf{x}_{1:T}, \mathbf{A}) \Big]$$
$$\approx \frac{1}{N} \sum_{n=1}^{N} q_\phi(\mathbf{z}_{1:L}|\mathbf{x}_{1:T}^{(n)}, \mathbf{A}). \quad (24)$$

In practice we approximate this quantity with

$$q_\phi^*(\mathbf{z}_{1:L}|\mathbf{A}) \approx q_\phi^*(\mathbf{z}_1) \prod_{i=2}^{L} q_\phi^*(\mathbf{z}_i|\mathbf{z}_{i-1}, \mathbf{A}), \quad (25)$$

where $q_\phi^*(\mathbf{z}_1)$ is a categorical distribution whose class probabilities $\rho_j^*(\phi)$ are the average of those from our approximate posterior, that is

$$\rho_j^*(\phi) = \frac{1}{N} \sum_{n=1}^{N} \rho_j(\mathbf{x}_{1:T}^{(n)}, \phi), \qquad (26)$$

and the transition probabilities $q_\phi^*(\mathbf{z}_i|\mathbf{z}_{i-1}, \mathbf{A})$ have transition probability matrices

$$Q_{k,j}^{*[i]}(\mathbf{A}, \phi) = \frac{1}{N} \sum_{n=1}^{N} Q_{k,j}^{[i]}(\mathbf{x}_{1:T}^{(n)}, \mathbf{A}, \phi). \quad (27)$$

D.4 Training Objective

Putting everything together, the training objective for the HSN model reads

$$\mathcal{L}[\theta, \phi] = \frac{1}{N} \sum_{n=1}^{N} \Big\{$$
$$\mathbb{E}_{q_\phi(\mathbf{z}_{1:L}|\mathbf{x}_{1:T}^{(n)}, \mathbf{A}) q_\phi(\mathbf{A})} \log p_\theta(\mathbf{x}_{1:T}^{(n)}|\mathbf{z}_{1:L}) \Big\}$$
$$- \mathbb{E}_{q_\phi(\mathbf{A})} \text{KL}\Big[q_\phi^*(\mathbf{z}_{1:L}|\mathbf{A}); p(\mathbf{z}_{1:L}|\mathbf{A}) \Big]$$
$$- \text{KL}[q_\phi(\mathbf{A}); p(\mathbf{A})]. \quad (28)$$

D.5 Mean-Field Solution

Instead of modeling the posterior over random walks in the usual way, we could consider a mean-field decomposition along the time component, by ignoring the dependency on the graph $\mathcal{G}$

$$q_\phi(\mathbf{z}_{1:L}|\mathbf{x}_{1:T}) = \prod_{i=1}^{L} q_\phi(\mathbf{z}_i|\mathbf{x}_{1:T}), \qquad (29)$$

where at each step of the walk we have a step-dependent categorical distribution

$$q_\phi(\mathbf{z}_i|\mathbf{x}_{1:T}) = \prod_{j=1}^{K} \Big(\rho_j^{[i]}(\mathbf{x}_{1:T}, \phi) \Big)^{z_i^j}, \qquad (30)$$

whose class probabilities live in the K-simplex. We could model the latter via

$$\boldsymbol{\rho}^{[1]}, \dots, \boldsymbol{\rho}^{[L]} = \text{softmax}(\mathbf{h}_1^{\text{enc}}, \dots, \mathbf{h}_L^{\text{enc}}) \quad (31)$$

where $\mathbf{h}_1^{\text{enc}}, \dots, \mathbf{h}_L^{\text{enc}}$ are the outputs of our encoder neural network model, shown in Figure 1 of the main text.

Replacing the mean-field approximation of

	Dataset	n. edges	$\mathcal{D}$	l	$\mathcal{C}$	$\mathcal{CC}$	largest $\mathcal{CC}$
HSN(50, 5)	PTB	694.26 ± 9.47	2.00 ± 0.00	1.43 ± 0.01	0.83 ± 0.01	1.00 ± 0.00	50.00 ± 0.00
	YAHOO	892.67 ± 8.22	2.00 ± 0.00	1.24 ± 0.01	0.84 ± 0.00	2.00 ± 0.04	49.00 ± 0.04
	YELP	891.06 ± 6.50	2.73 ± 0.46	1.24 ± 0.03	0.84 ± 0.01	2.24 ± 0.85	48.76 ± 0.85
	Random	611.69 ± 17.61	2.00 ± 0.00	1.50 ± 0.01	0.50 ± 0.02	1.00 ± 0.00	50.00 ± 0.00
HSN(50, 20)	PTB	764.28 ± 7.88	2.83 ± 0.38	1.27 ± 0.03	0.82 ± 0.01	4.71 ± 0.77	46.29 ± 0.77
	YAHOO	356.35 ± 7.76	3.17 ± 0.37	1.57 ± 0.05	0.58 ± 0.02	12.04 ± 1.49	38.96 ± 1.49
	YELP	259.42 ± 5.47	2.68 ± 0.48	1.42 ± 0.03	0.48 ± 0.01	20.77 ± 0.68	30.23 ± 0.68
	Random	611.69 ± 17.61	2.00 ± 0.00	1.50 ± 0.01	0.50 ± 0.02	1.00 ± 0.00	50.00 ± 0.00
HLN(100, 5)	PTB	1198.18 ± 16.44	2.56 ± 0.50	1.76 ± 0.01	0.83 ± 0.01	1.19 ± 0.40	99.81 ± 0.40
	YAHOO	1239.21 ± 12.19	3.15 ± 0.38	1.42 ± 0.03	0.51 ± 0.01	35.93 ± 1.41	65.07 ± 1.41
	YELP	1295.68 ± 12.93	3.36 ± 0.48	1.55 ± 0.03	0.57 ± 0.01	27.38 ± 1.69	73.62 ± 1.69
	Random	2474.92 ± 36.58	2.00 ± 0.00	1.50 ± 0.01	0.50 ± 0.01	1.00 ± 0.00	100.00 ± 0.00
HLN(100, 20)	PTB	892.53 ± 10.04	3.04 ± 0.24	1.41 ± 0.01	0.45 ± 0.01	46.04 ± 1.53	54.96 ± 1.54
	YAHOO	261.13 ± 7.14	2.18 ± 0.38	1.95 ± 0.00	0.91 ± 0.01	1.01 ± 0.10	99.99 ± 0.10
	YELP	515.84 ± 10.09	3.68 ± 0.48	1.79 ± 0.06	0.38 ± 0.02	45.27 ± 2.58	55.67 ± 2.58
	Random	2474.92 ± 36.58	2.00 ± 0.00	1.50 ± 0.01	0.50 ± 0.01	1.00 ± 0.00	100.00 ± 0.00

Table 4: Statistic of inferred graphs for all datasets

Eq. 29 into Eq. 10 yields

$$
\begin{aligned}
&\mathrm{KL}L[q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}^{(n)}); p(\mathbf{z}_{1:T}|\mathbf{A}) \\
&= \sum_{i=2}^{L} \Big\{ \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_{1:T}^{(n)})} \log q_\phi(\mathbf{z}_i|\mathbf{x}_{1:T}^{(n)}) \\
&\quad - \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_{1:T}^{(n)})q_\phi(\mathbf{z}_{i-1}|\mathbf{x}_{1:T}^{(n)})} \log p(\mathbf{z}_i|\mathbf{z}_{i-1}) \Big\} \\
&\quad + \mathrm{KL}[q_\phi(\mathbf{z}_1); p(\mathbf{z}_1)], \\
&= \sum_{i=1}^{L} \sum_{j}^{K} \rho_j^{[i]}(\mathbf{x}_{1:T}, \phi) \log \frac{\rho_j^{[i]}(\mathbf{x}_{1:T}, \phi)}{\rho_j} \\
&\quad - \sum_{i=2}^{L} \sum_{k,j}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_{1:T}^{(n)})q_\phi(\mathbf{z}_{i-1}|\mathbf{x}_{1:T}^{(n)})} \Big[z_i^k z_{i-1}^j \Big] \\
&\quad \times \log P_{k,j} \\
&= \sum_{i=1}^{L} \sum_{j}^{K} \rho_j^{[i]}(\mathbf{x}_{1:T}, \phi) \log \frac{\rho_j^{[i]}(\mathbf{x}_{1:T}, \phi)}{\rho_j} \\
&\quad - \sum_{i=2}^{L} \sum_{k,j}^{K} \rho_k^{[i]}(\mathbf{x}_{1:T}, \phi) \rho_j^{[i-1]}(\mathbf{x}_{1:T}, \phi) \\
&\quad \times \log P_{k,j}. \quad (32)
\end{aligned}
$$

D.6 Fully Connected Graph

We can replace the adjacency matrix $\mathbf{A}$ in the definition of the transition probability matrix of our posterior $\mathbf{Q}(\mathbf{x}_{1:T}, \mathbf{A}, \phi)$, with that of a fully connected graph. The aggregated posterior over all walks up to step i (Eq. 12 above) reduces in this case to

$$
\begin{aligned}
&\hat{\rho}_k^{[i]}(\mathbf{x}_{1:T}, \phi) = \\
&\sum_{j}^{K} \left(\frac{f_k^{[i-1]}(\mathbf{x}_{1:T}, \phi) A_{k,j}}{\sum_m f_m^{[i-1]}(\mathbf{x}_{1:T}, \phi) A_{m,j}} \right) \hat{\rho}_j^{[i-i]}(\mathbf{x}_{1:T}, \phi) \\
&= \left(\frac{f_k^{[i-1]}(\mathbf{x}_{1:T}, \phi)}{\sum_m f_m^{[i-1]}(\mathbf{x}_{1:T}, \phi)} \right) \left(\sum_{j}^{K} \hat{\rho}_j^{[i-i]}(\mathbf{x}_{1:T}, \phi) \right) \\
&= \frac{f_k^{[i-1]}(\mathbf{x}_{1:T}, \phi)}{\sum_m f_m^{[i-1]}(\mathbf{x}_{1:T}, \phi)}, \quad (33)
\end{aligned}
$$

which is equivalent to that of the mean-field approximation of section D.5 with $\hat{\rho}_k^{[i]} = \rho_k^{[i]}$.

E On Synthetic Dataset Experiments

In this section we give additional details of and results from our proof-of-concept experiments.

E.1 Synthetic Language Model

We generate our synthetic dataset as follows: first, we sample a single, fixed graph $\mathcal{G}^*$ with K nodes from a predefined random graph model. Second, we define a set of random tokens $\mathcal{V}$, of size V, to be our vocabulary. We create each token as a random 3-tuple from the Latin alphabet, and choose to have at least one order of magnitude more tokens than nodes in G (that is, $V \gg K$). Third, we assign a random bag of tokens to each node in $\mathcal{G}^*$. These random bags can simply be understood as probability distributions over $\mathcal{V}$, and can be represented as V-dimensional vectors whose components live on the simplex. Note in particular that, by construction, tokens can be shared among the different nodes of $\mathcal{G}^*$. Finally, let us identify the K random bags with the K symbols $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_K\}$ of the synthetic language model.

To generate synthetic sentences we sample uniform, L-step random walks on $\mathcal{G}^*$, whose transition matrix is given by Eq. 4 in the main text, with $\mathbf{f} = \mathbb{I}$. Having obtained a set of random walks on $\mathcal{G}^*$, we sample one random token from each of the symbols (i.e. from each random bag) along the walks.

E.2 Experimental Settings

Here we give additional details for reproducibility

Datasets

- Following the procedure above we generated two datasets from two random graphs with different topologies. One sampled from the Barabási-Albert model (Barabási and Albert, 1999), the other from the Erdös-Rényi model (Erdös and Rényi, 1959). We generate these graphs using NetworkX, a Python language software package for network structures (Hagberg et al., 2008). Specifically, we generate Barabási-Albert graphs by attaching 3 edges from each new node to old ones, and Erdös-Rényi graphs with an edge probability of 0.5. We set both graphs to have $K = 100$ symbols.

- We define each random bag of tokens in $\mathcal{G}^*$ to have two tokens only (each with equal probability).

- We use a vocabulary of 1000 random tokens.

- Once the graph is fixed, we set the token sequence length to $L = 10$ ($L = 11$) for the Erdös (Barabási) datasets and generate a total of $N = 100000$ token sequences from each random graph.

Hidden Schema Network (HSN) **settings**

- We train randomly initialized embeddings of dimension 256, one for each token. We sample these from a normal distribution with zero mean and a standard deviation of 0.01.

- The posterior graph model is defined via a single feed-forward neural network with 256 hidden units.

- The prior graph model has the edge probability p as hyperparameter. We crossvalidate it from the set $p = \{0.1, 0.2, 0.5, 0.6, 0.8\}$ and found that HSN could fit the Barabási dataset only with small values $\{0.1, 0.2\}$. HSN could fit the Erdös dataset with larger values $\{0.5, 0.6\}$

- The posterior random walk model is defined by replacing BERT with a 2-block Transformer encoder (Vaswani et al., 2017), each with 2 heads, 256 hidden units and dropout probability of 0.2.

- The prior random walk model was set to a uniform random walk.

Training details

- We use a batch size of 256 and train with Adam (Kingma and Ba, 2014), with a learning rate of 0.0001, in all experiments.

- To sample both graph and random walk posterior models with use the Gumbel-Softmax trick (Jang et al., 2016), with a constant temperature of 0.75

- We train the models for 200 epochs

E.3 Additional Results

Table 5 displays the mean and standard deviation of some additional results on our proof-of-concept experiments. We trained ten models in total.

We first trained a simple LSTM Network to infer the correct symbol order in each random token sequence. We noticed that a network with 256 hidden units was enough to solve this task perfectly. Indeed, the negative log-likelihood (NLL) of these models corresponds to choosing the 2-token random bag sequence (i.e. the schema) that yields the correct token sequence without errors. The HSN performs equally well on the Barabási dataset, and slightly worst on the Erdös dataset. In fact, we have noticed the Erdös dataset proved to be more challenging to learn with the HSN in all regards. See, for example, the AUC scores or the Frobenious norms of HSN in this dataset, as compared to the Barabási case. We think this might be due to the fact that Barabási graphs have more structure, simply because of their sparsity, which arguably make them easier to infer with our inductive bias.

Note also how increasing the prior edge probability p affects the average number of edges of the inferred graphs.

F On Language Modelling Experiments

In this section we give additional details of and results from our language modelling and representation learning experiments.

Graph $\mathcal{G}^*$	Model	NLL	KL $-$ $\mathbf{z}$	KL $-$ $\mathcal{G}$	AUC	$|\mathcal{G}^* - \mathcal{G}|_F$	$|\mathcal{G}^r - \mathcal{G}|_F$	N. edges($\mathcal{G}$)
	LSTM	**53.07± 0.01**	–	–	–	–	–	–
Barabasi	HS (0.1)	53.08 ± 0.01	0.10 ± 0.06	9 ± 1	0.977 ± 0.003	17 ± 2	27 ± 1	1090 ± 143
	HS (0.2)	53.07 ± 0.02	0.09 ± 0.06	4.8 ± 0.5	0.989 ± 0.001	17 ± 2	26 ± 1	1360 ± 104
	LSTM	**48.24 ± 0.02**	–	–	–	–	–	–
Erdos	HS (0.5)	50.9 ± 0.8	1.2 ± 0.3	4 ± 6	0.95 ± 0.06	34.8 ± 0.9	40 ± 5	2812 ± 344
	HS (0.6)	50.4 ± 0.6	1.3 ± 0.1	1 ± 2	0.94 ± 0.06	36.8 ± 0.8	44 ± 2	3131 ± 156

Table 5: Inference on ground-truth random graphs. Here we use the notation HS(p) to denote Hidden Schema Network models with prior graph distributions whose edge probability is set to p.

F.1 Experimental Settings

Here we give additional details for reproducibility

Datasets

- We consider three widely used public datasets, namely the Penn Treebank (PTB) (Marcus et al., 1993), Yahoo and Yelp (Yang et al., 2017) corpora.

- PTB training set has a total of 38219 sentences. The average length of which is of about 22 words. The validation and test set have 5527 and 5462 sentences, respectively. The minimum (maximum) sentence length in PTB is of 2 (78) words.

- Yahoo training set has a total of 100000 sentences. The average length of which is of about 80 words. The validation and test sets have 10000 sentences each. The minimum (maximum) sentence length in Yahoo is of 21 (201) words. The Vocabulary size is of 200000 words.

- Yelp training set has a total of 100000 sentences. The average length of which is of about 97 words. The validation and test sets have 10000 sentences each. The minimum (maximum) sentence length in Yelp is of 21 (201) words. The Vocabulary size is of 90000 words.

HSN settings

- In all experiments we leveraged pretrained BERT and GPT-2 models, both with 12 layers, 768 hidden dimensions (D) and 12 attention heads. We used the public HuggingFace implementation of both these models (Wolf et al., 2020).

- The posterior graph model is set to a 2-layer feed forward network, each with hidden dimension 512.

- We crossvalidated the prior edge probability over the set of values $p = \{0.1, 0.2, 0.5, 0.6\}$ and found $p = 0.5$ (a maximum entropy prior) to yield the best results. All results we report correspond to this ($p = 0.5$) case.

- We also train an inhomogeneous random walk prior model by making $\boldsymbol{\rho}$ and the sequence of weights $\mathbf{f}^{[1]}, \mathbf{f}^{[2]}, \ldots, \mathbf{f}^{[L-1]}$ trainable. We initialized them by sampling from a normal distribution with zero mean and standard deviation of 0.01.

- We experimented with HSN of $K = \{50, 100\}$ symbols and random walks of length $L = \{5, 20\}$.

Training details

- We used a batch size of 32 and train with Adam (Kingma and Ba, 2014), with a learning rate of 0.00001, in all experiments.

- To sample both graph and random walk posterior models with used the Gumbel-Softmax trick (Jang et al., 2016), with a constant temperature of 1.0.

- We used a cyclical schedule to anneal both KL terms in our training objective from zero to one (Fu et al., 2019). When the annealing weight (usually called β in the literature) is finite, we used a KL threshold scheme (Li et al., 2019), with a threshold value of 0.1.

- We trained the models for 100 epochs, although they usually needed about 60 epochs only to converge (in the NLL).

- We applied word dropout to the input of the decoder model with probability 0.3 in the following cases: (i) for all models trained on PTB; (ii) and all models with $L = 50$ trained on all datasets.

F.2 Evaluation Metrics

Monte Carlo Perplexity Estimation

We compute the perplexity of HSN models via the importance weighted Monte Carlo estimator of the marginal log-likelihood,

$$\log p_\theta(\mathbf{x}_{1:T}) = \log \frac{1}{R} \sum_r^R \frac{1}{S} \sum_s^S$$

$$\frac{p_\theta(\mathbf{x}_{1:T}|\mathbf{z}_{1:L}^{(r)})p(\mathbf{z}_{1:L}^{(r)}|\mathbf{A}^{(s)})p(\mathbf{A}^{(s)})}{q_\phi(\mathbf{z}_{1:L}^{(r)}|\mathbf{A}^{(s)})q_\phi(\mathbf{A}^{(s)})}, \quad (34)$$

where R and S are the number of random walk samples and graph samples, respectively. We used $R = 100$ and $S = 10$ per each sentence in the test set to estimate the perplexity.

Mutual Information

Another metric that we use to evaluate our models is the mutual information between the representations (random walks) and the word sequences

$$I(\mathbf{z}; \mathbf{x}) =$$
$$\mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})} \mathrm{KL}[q(\mathbf{z}|\mathbf{x}); p(\mathbf{z})] - \mathrm{KL}[q^*(\mathbf{z}); p(\mathbf{z})]. \quad (35)$$

Full derivation of this expression can be found in Appendix D.3 above.

F.3 Additional Results

Here we report results complementing the conclusions of the main text.

Language modelling. Table 6 displays mutual information, reconstruction loss, KL of the random walks and KL of the graph. We report their mean and standard deviation obtained when repeating the experiments with the HSN model five times, with different initializations.

The conclusion of the main text, viz. that our results outperform all baselines, remains unaltered, even within error bars.

Graph statistics. We characterize the structure of $\mathcal{G}$ in terms of five statistics: (i) the diameter $\mathcal{D}$, which measures the maximum path length over all node pairs in $\mathcal{G}$; (ii) the average distance l, which instead measures the average shortest path length between all node pairs; (iii) the clustering coefficient $\mathcal{C}$, which represents the probability that two neighbors of a randomly chosen node are themselves neighbors; (iv) the number of connected components $\mathcal{CC}$; and (v) the degree distribution

$P(k)$, which represents the probability that a randomly chosen node will have k neighbors.

Table 4 reports the statistics of our inferred graphs for all datasets, and all model configurations.

We can see that increasing the random walk length from 5 to 20 increases the number of connected components of the graphs. As a consequence, subsets of word sequences are map onto smaller subgraphs, the larger of which is about 50 symbols. One could argue that, since longer random walk lengths imply a larger set of possible schema configurations, the number of symbols required to describe our three corpora can simply decrease. In other words, less symbols are needed by long schemata. Similarly, directly increasing the symbols number leads too to a larger number of connected components. Indeed, even the short schemata in Yelp and Yahoo do not use all available symbols to model the corpora.

Finally, Figure 11 displays the difference between the degree distributions of the schema networks, and that of an Erdös-Rényi graph for all HSN configurations.

Schema distributions on inferred networks. We can get a graphical picture of the features we just discussed above in Figures 4–6 below. Importantly, we see that the schema distribution is different for each category of each corpora in all model configurations. In other words, we do not observe any kind of mode collapse.

Schema interpolations. Finally, we have also explored "schema interpolations": given two schemata $e_{j_1:j_L}$ and $e_{m_1:m_L}$, we find the shortest path (of length l) on $\mathcal{G}$ connecting the end of $e_{j_1:j_L}$ with the beginning of $e_{m_1:m_L}$. Our interpolation steps are the schemata $\{e_{j_1+i:j_L+i} : \forall\, 0 \leq i \leq l + L$ along the path$\}$. Tables 9–10 show interpolations of random instances from all datasets. Note how the model successfully interpolates between categories in both Yelp and Yahoo.

G Which Symbols do Words Attend to? A Preliminary Study on Yelp Reviews

In this section we investigate how symbols are used by HSN when generating text. We do this by exploring the decoder attention matrix between the symbols and the generated tokens. Reading the attention weights, we can examine which symbols are most important for the generation of any given token, i.e. which symbols are attended to more

	Model	MI	Rec	KL	$KL_{\mathcal{G}}$
	iVAE$_{MI}$	12.50	74.69	12.51	–
	Optimus$_A$	3.78	86.43	4.88	–
	Optimus$_B$	8.18	77.65	28.50	–
PTB	HSN$_{50,20}$	25.3 ± 0.9	77.1 ± 0.4	27 ± 1	0.1 ± 0.2
	HSN$_{50,5}$	9.3 ± 0.2	78.7 ± 0.2	10.3 ± 0.2	0.02 ± 0.01
	HSN$_{100,20}$	23 ± 1	77.9 ± 0.3	24 ± 1	0.21 ± 0.05
	HSN$_{100,5}$	9.1 ± 0.3	79.1 ± 0.2	10.2 ± 0.3	0.11 ± 0.02
	iVAE$_{MI}$	10.70	297.70	11.40	–
	Optimus$_A$	5.34	282.84	6.97	–
	Optimus$_B$	9.18	270.80	30.41	–
YAHOO	HSN$_{50,20}$	25 ± 2	267 ± 1	26 ± 3	0.07 ± 0.1
	HSN$_{50,5}$	10.2 ± 0.4	268.4 ± 0.4	11.4 ± 0.5	0.08 ± 0.04
	HSN$_{100,20}$	13 ± 5	268 ± 1	13 ± 5	0.3 ± 0.1
	HSN$_{100,5}$	10.8 ± 0.4	267.7 ± 0.5	12.3 ± 0.6	0.04 ± 0.04
	iVAE$_{MI}$	11.00	348.70	11.60	–
	Optimus$_A$	2.54	334.31	3.09	–
	Optimus$_B$	9.13	325.77	27.89	–
YELP	HSN$_{50,20}$	24 ± 2	312 ± 3	26 ± 3	0.1 ± 0.1
	HSN$_{50,5}$	10.0 ± 0.2	312.0 ± 0.5	11.3 ± 0.2	0.025 ± 0.009
	HSN$_{100,20}$	18 ± 3	312.7 ± 0.6	17 ± 3	0.3 ± 0.1
	HSN$_{100,5}$	10.1 ± 0.2	312.1 ± 0.5	11.5 ± 0.3	0.16 ± 0.02

Table 6: Mutual information (MI), reconstruction loss (Rec), KL of the random walks (KL) and KL of the graph (KL$_{\mathcal{G}}$) for iVAE. Optimus$_{A, B}$ label models with best PPL and MI, respectively (with $\lambda = 0.05, 1$) (Li et al., 2020) . iVAE$_{MI}$ was taken from Fang et al. (2019). For Hidden Schema Network models HSN$_{K,L}$ with K symbols and random walk length L, we report mean and standard deviation obtained by repeating each experiment five times with different initializations.

layer	KL (good, bad)	KL (good, great)	KL (great, bad)
1	0.807	0.336	1.227
5	0.738	0.177	1.245
12	0.635	0.224	0.957

Table 7: Kullback-Leibler divergence between the distributions of most attended symbols, when generating the tokens *good*, *bad* and *great*. Results are computed with HSN(100, 5) trained on Yelp. The KL values are computed for each head separately and then averaged.

strongly. A bit more in detail we select, for a given token in a given sentence, the symbol with the highest attention value. We can then compute the distribution of most attended symbols when generating that token for the complete dataset.

Thus, for a model trained on the Yelp dataset, we examine to which symbols does the decoder of HSN attend to, when processing the words *good*, *great* and *bad*. Figure 7 shows the most attended symbol distribution for layers 1 (first), 5 (middle), 12 (last), when averaging the attention matrices over all attention heads. Figures 8, 9, 10 show

these distributions for each head separately. Note how, for a fixed token, the distribution of attention changes as one moves between heads and layers, although there are some repeating patterns too.

We can quantify these features by computing the Kullback-Leibler (KL) divergence between these distributions. The KL values are shown in Table 7.

Interestingly enough, the distribution of symbols that are attended to when processing the word *great* is closer to the distributions of symbols attended by the word *good*, than to the distributions of symbols attended by the word *bad*.

H On Commonsense Reasoning Generation

In this section we expatiate on the details of our approach to commonsense reasoning generation.

First, we modify the encoder component of HSN to process the tuples $(\mathbf{s}, \mathbf{r}, \mathbf{o})$ as

$$q_\phi(\mathbf{z}_{1:L}|[\mathbf{s}, \mathbf{r}, \mathbf{o}], \mathbf{A}) =$$
$$q_\phi(\mathbf{z}_{1:\frac{L}{2}}|[\mathbf{s}, \mathbf{r}], \mathbf{A})q_\phi(\mathbf{z}_{\frac{L}{2}+1:L}|[\mathbf{s}, \mathbf{r}, \mathbf{o}], \mathbf{z}_{\frac{L}{2}}, \mathbf{A}),$$

so that the first half of the schema depends on subject and relation only, whereas the second half depends on the entire 3-tuple. As it will become evident below, this decoupling is necessary for the inference of novel objects.

Each of the posterior distributions above is modelled with the same architecture, as shown in Fig. 1, but sharing a single pretrained BERT model. That is, we have two copies of all pink-shaded blocks in the Fig. 1, one for $q_\phi(\mathbf{z}_{1:\frac{L}{2}}|[\mathbf{s},\mathbf{r}],\mathbf{A})$, the other for $q_\phi(\mathbf{z}_{\frac{L}{2}+1:L}|[\mathbf{s},\mathbf{r},\mathbf{o}],\mathbf{z}_{\frac{L}{2}},\mathbf{A})$, and a single pretrained BERT model. Using such a 2-component encoder model we are able to successfully infer schema representations for the KG tuples, as shown in Table 3. The task is however to infer *new objects*, given only subject-relation pairs. We thus need a way to infer schema representations without relying on the phrase object.

The classical solution to this inference problem is to replace, à la Kalman Filter, $q_\phi(\mathbf{z}_{\frac{L}{2}+1:L}|[\mathbf{s},\mathbf{r},\mathbf{o}],\mathbf{z}_{\frac{L}{2}},\mathbf{A})$ with a local, trainable prior model of the form $p_\theta(\mathbf{z}_{\frac{L}{2}+1:L}|[\mathbf{s},\mathbf{r}],\mathbf{z}_{\frac{L}{2}},\mathbf{A})$ – where $\mathbf{z}_{\frac{L}{2}}$ is sampled from $q_\phi(\mathbf{z}_{1:\frac{L}{2}}|[\mathbf{s},\mathbf{r}],\mathbf{A})$ – and train the prior via the KL term in Eq. 28. However, and as shown in Section D, maximizing the mutual information between data and representations averages out all local information in the KL term, and thus hinders the learning of the prior.

An alternative is to train, in the spirit of knowledge distillation (Hinton et al., 2015), a third-party model on the inferred schemata, to predict $\mathbf{z}_{\frac{L}{2}+1:L}$ conditioned on $\mathbf{z}_{1:\frac{L}{2}}$. Indeed, given the inferred schemata from the training KG, we consider a sequence-to-sequence model which inputs $\mathbf{z}_{1:\frac{L}{2}}$, together with the subject-relation pair, and outputs $\mathbf{z}_{\frac{L}{2}+1:L}$. That is, a model of the form $p_\theta(\mathbf{z}_{\frac{L}{2}+1:L}|\mathbf{z}_{1:\frac{L}{2}},[\mathbf{s},\mathbf{r}])$. Specifically we use (i) a bidirectional LSTM network with hidden dimension of 512 to encode the first half of the schemata, (ii) a pretrained BERT model to encode the subject-relation pair, and (iii) a LSTM network of dimension 512 as an autoregressive decoder model. The initial (hidden) states of the latter are determined by an MLP which inputs the representations from the LSTM and BERT encoder models. The model is trained on samples from $q_\phi(\mathbf{z}_{\frac{L}{2}+1:L}|[\mathbf{s},\mathbf{r},\mathbf{o}],\mathbf{z}_{\frac{L}{2}},\mathbf{A})$.

Our preliminary results, $\text{HSN}^{\text{AR}}_{(50,\,20)}$ in Table 3, are comparable to a stand-alone COMET(GPT-2) model, even when the thrid-party model only learns about 58% of the second half of the schemata.

H.1 Atomic Dataset

For this preliminary study we focus only on the ATOMIC dataset of Sap et al. (2019). It contains 877K $(\mathbf{s},\mathbf{r},\mathbf{o})$ tuples covering a variety of social commonsense knowledge around specific *If-Then* events. A bit more in detail, ATOMIC splits its commonsense knowledge into nine categories, covering the event's causes, its effects on the agent, and its effect on other direct (or implied) participants. We use the training splits from Sap et al. (2019), resulting in 710K training, 80K validation, and 87K test tuples respectively.

I Training Time and Resource Consumption

In Table 8 we report the training times of our HSN models. The times are similar for all HSN configurations, so we report their average training time for each dataset.
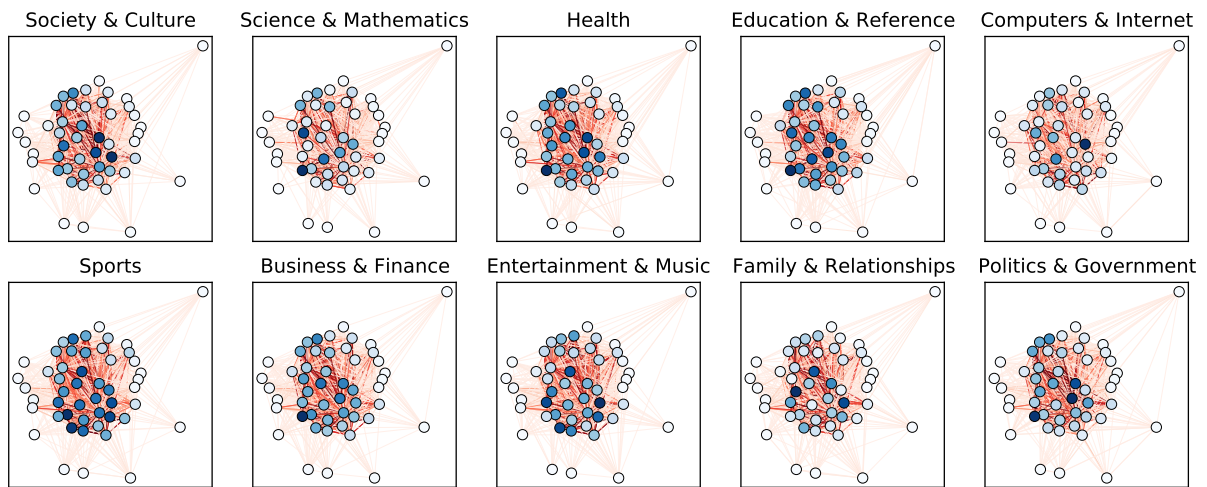
dataset	train time in hours	#sentences	avg words per sentence
Yahoo	22	100K	80
Yelp	26	100K	97
PTB	5	38K	22
Atomic	42	877	6

Table 8: Average training times of all HSN configurations per dataset. All HSN models have about 252M parameters. We ran each model on a single NVIDIA A100-SXM4-40GB GPU unit.
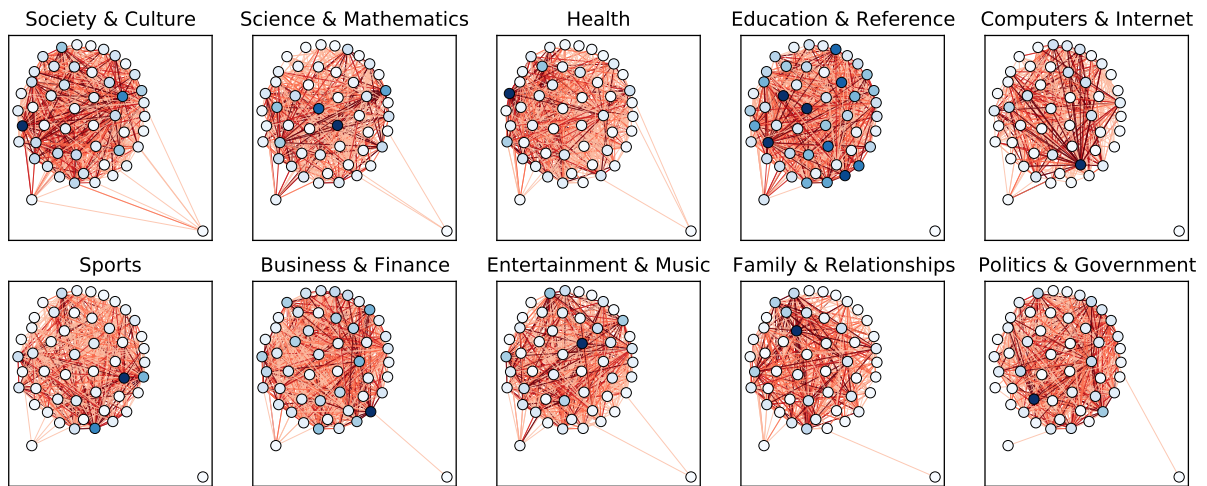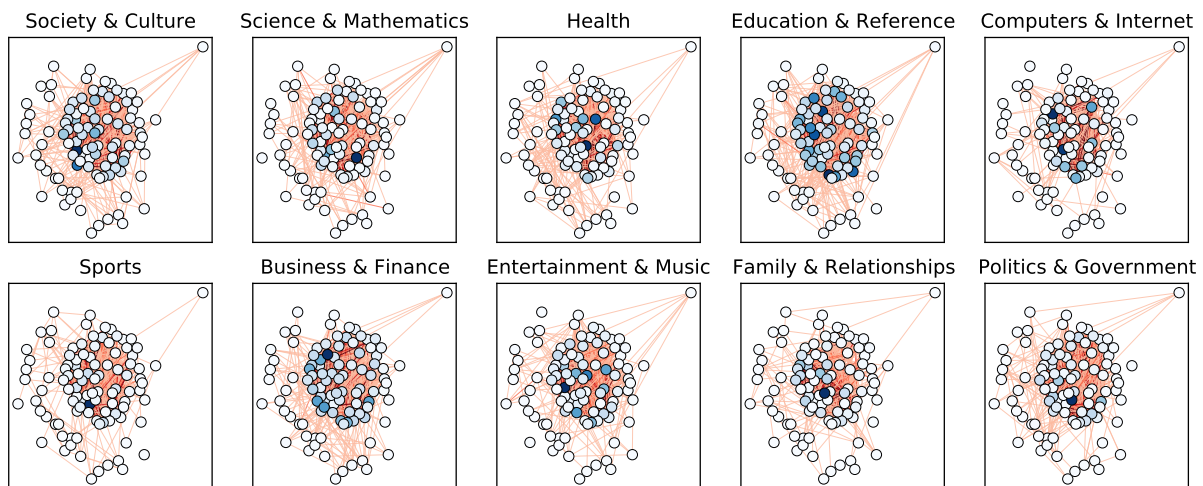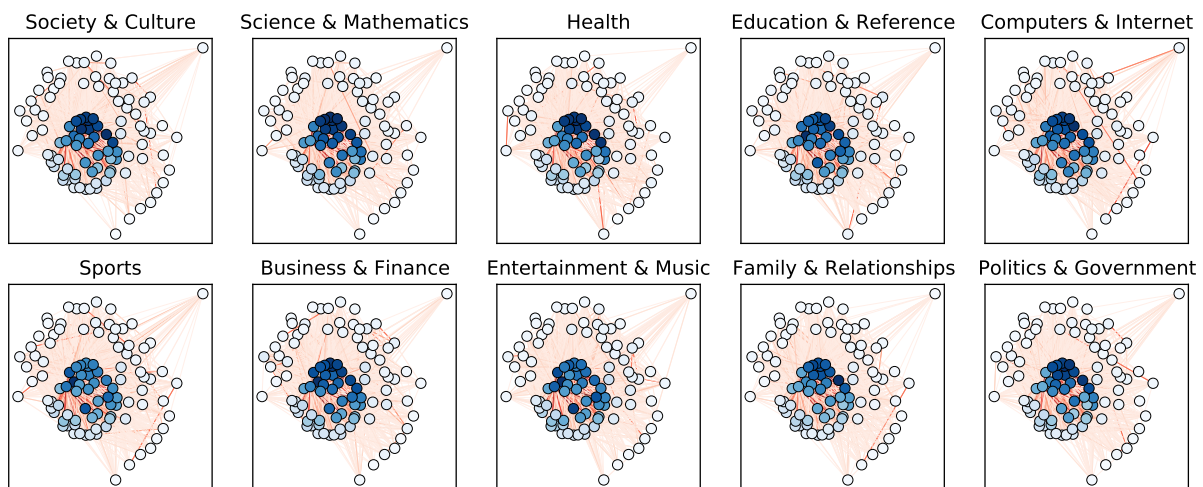
(a) HSN(50, 5)

(b) HSN(50, 20)

Figure 4: Schema distributions inferred from each category of the Yahoo dataset, for HSN(50, L) with $L = \{5, 20\}$. The node positions in the figure are consistent among labels and were computed using a force-directed embedding of the global graph $\mathcal{G}$.
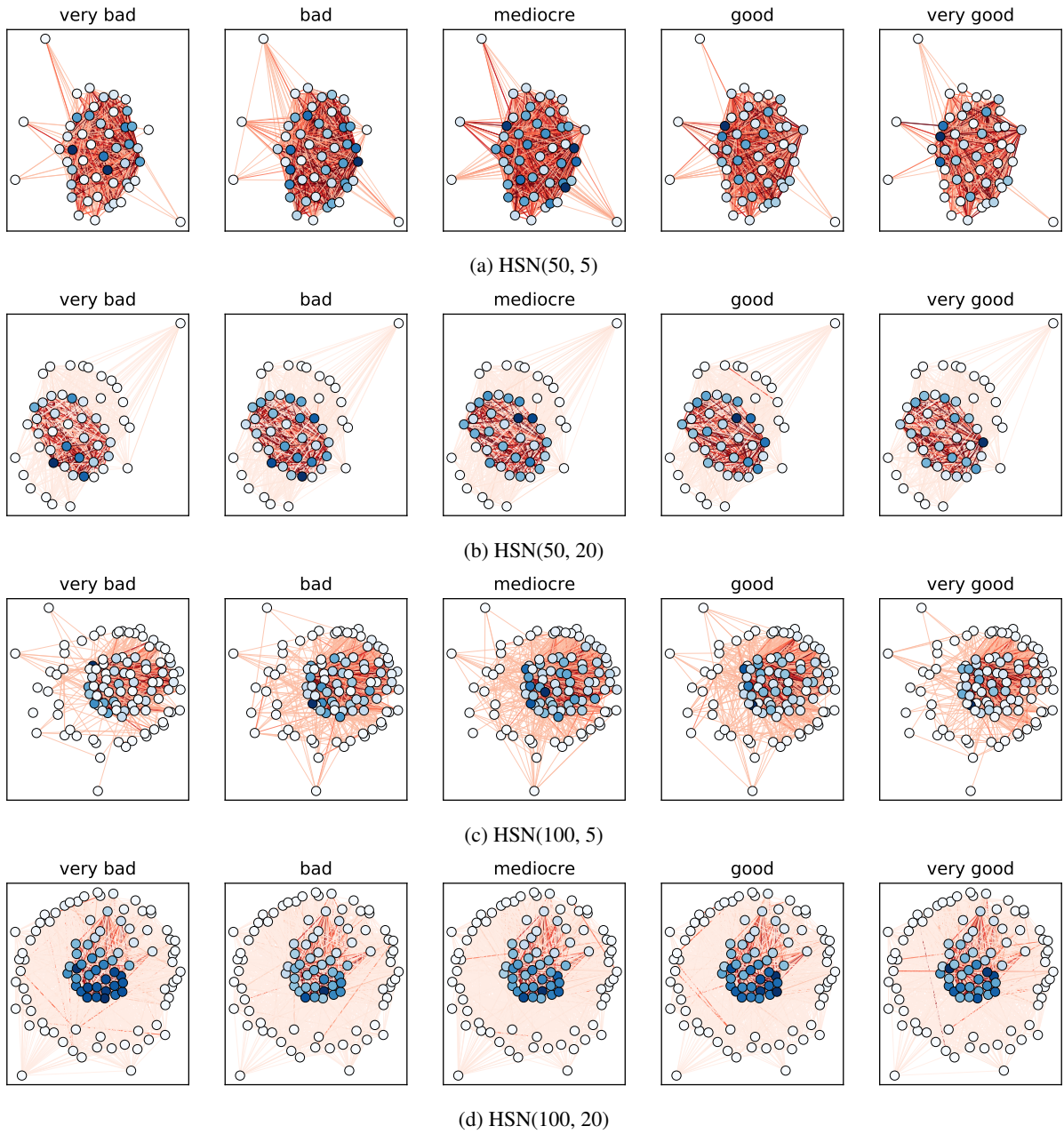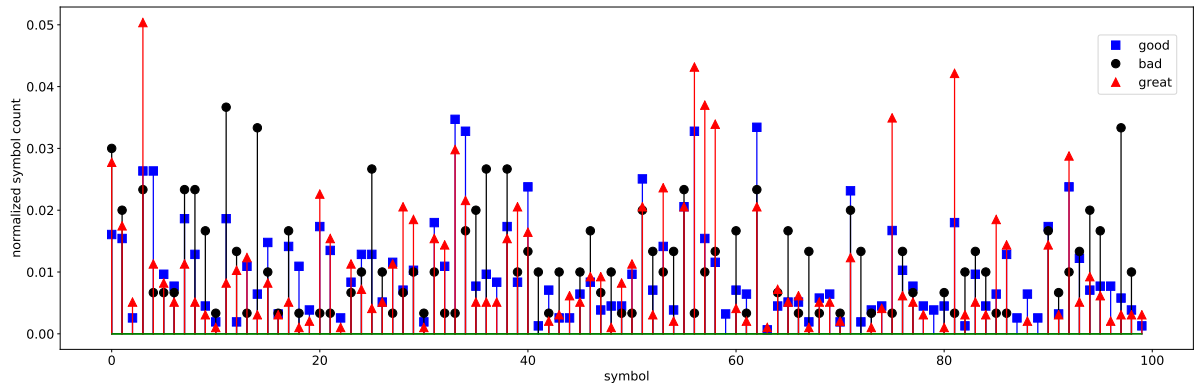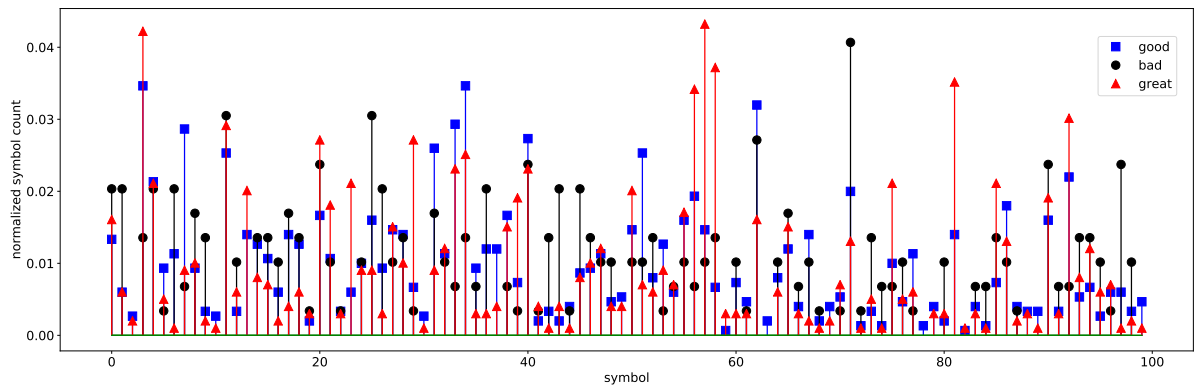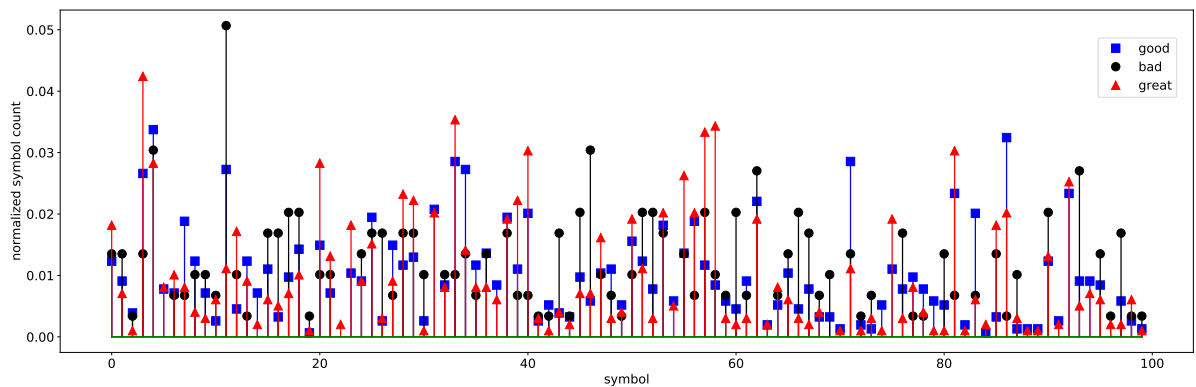
(a) HSN(100, 5)

(b) HSN(100, 20)

Figure 5: Schema distributions inferred from each category of the Yahoo dataset, for HSN(100, L) with $L = \{5, 20\}$. The node positions in the figure are consistent among labels and were computed using a force-directed embedding of the global graph $\mathcal{G}$.

Figure 6: Schema distributions inferred from each category of the Yelp dataset. The node positions in the figure are consistent among labels and were computed using a force-directed embedding of the global graph $\mathcal{G}$.

(a) Layer 1

(b) Layer 5

(c) Layer 12

Figure 7: Distribution of most attended symbols when generating tokens *good*, *bad*, *great* for HSN(100, 5) trained on the Yelp data set. The decoder attention matrices between symbols and output are averaged over all attention heads for layers 1, 5 and 12.
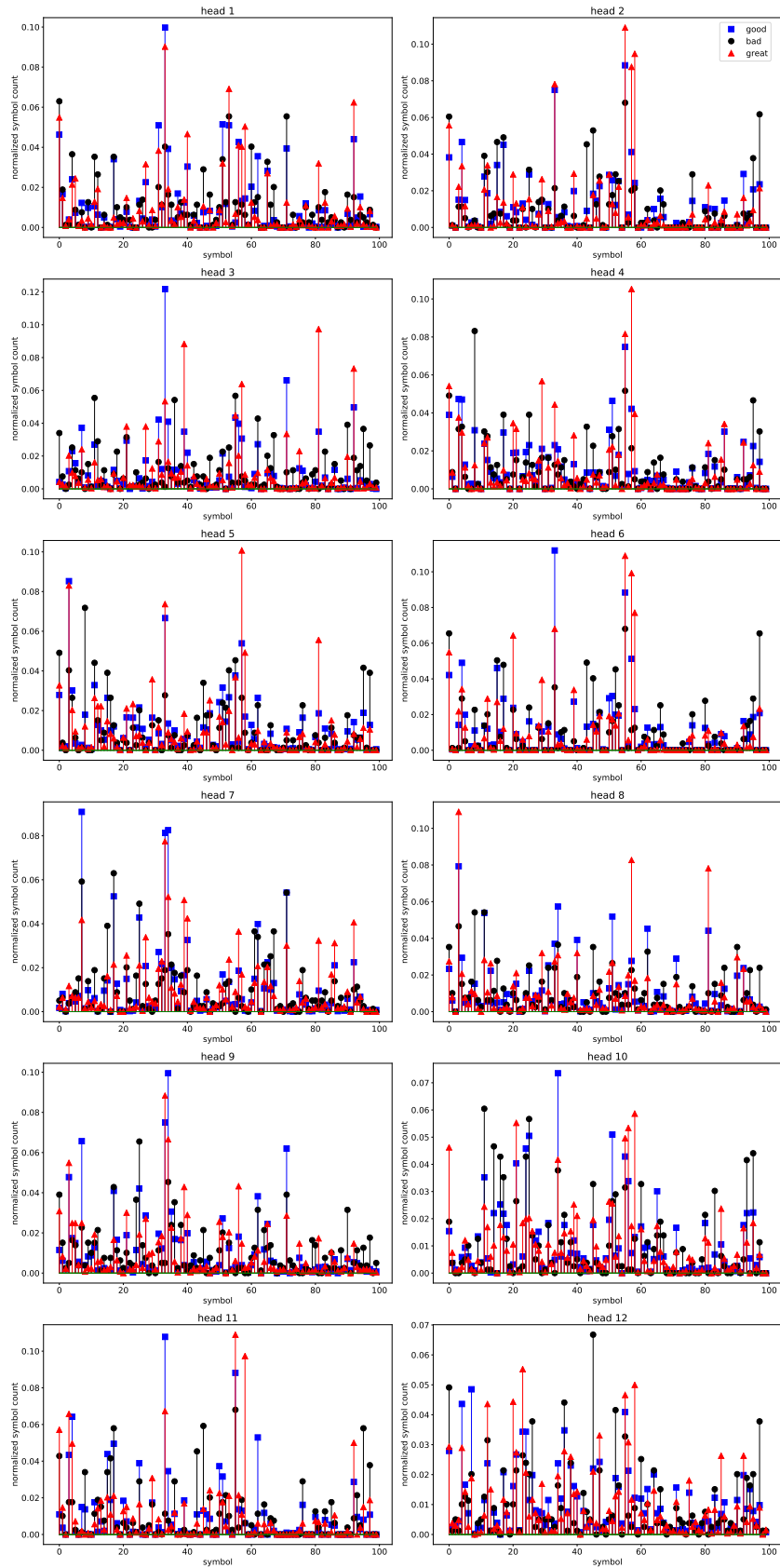
Figure 8: Distribution of most attended symbols when generating tokens *good*, *bad*, *great* for HSN(100, 5) trained on the Yelp data set. The distribution is computed from the decoder attention matrices between symbols and output for each attention head for layer 1.
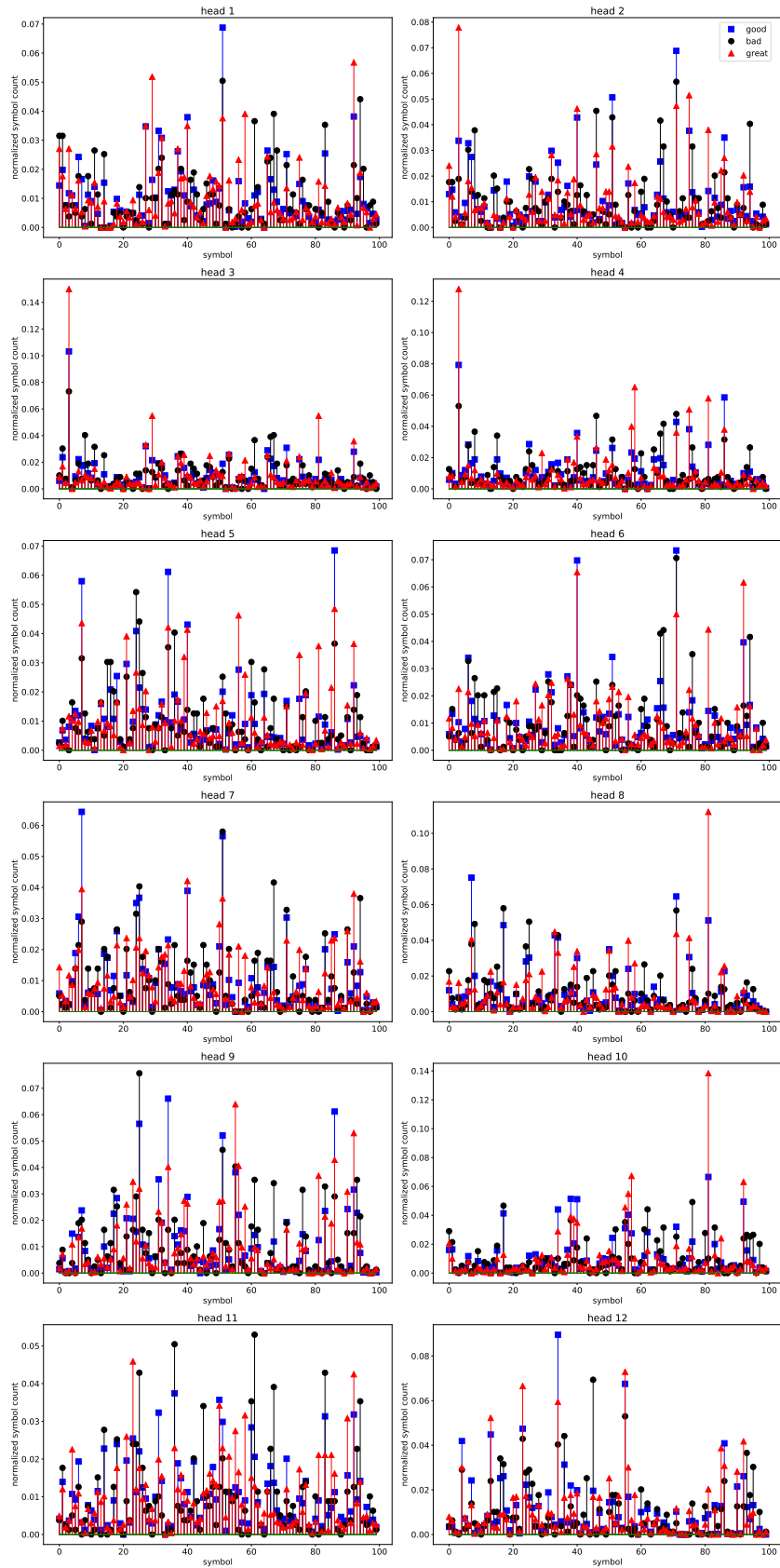
Figure 9: Distribution of most attended symbols when generating tokens *good*, *bad*, *great* for HSN(100, 5) trained on the Yelp data set. The distribution is computed from the decoder attention matrices between symbols and output for each attention head for layer 5.
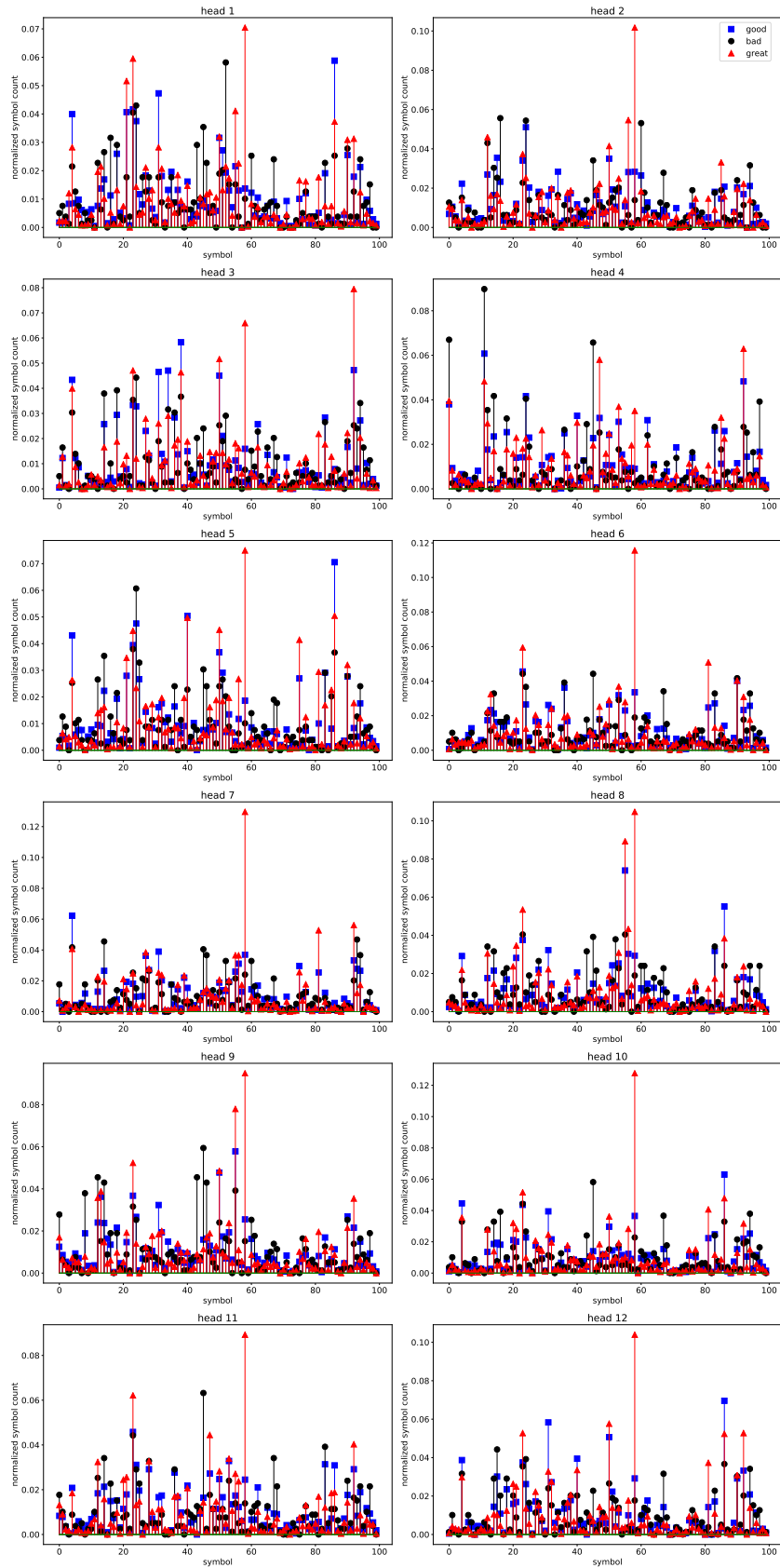
Figure 10: Distribution of most attended symbols when generating tokens *good*, *bad*, *great* for HSN(100, 5) trained on the Yelp data set. The distribution is computed from the decoder attention matrices between symbols and output for each attention head for layer 12.
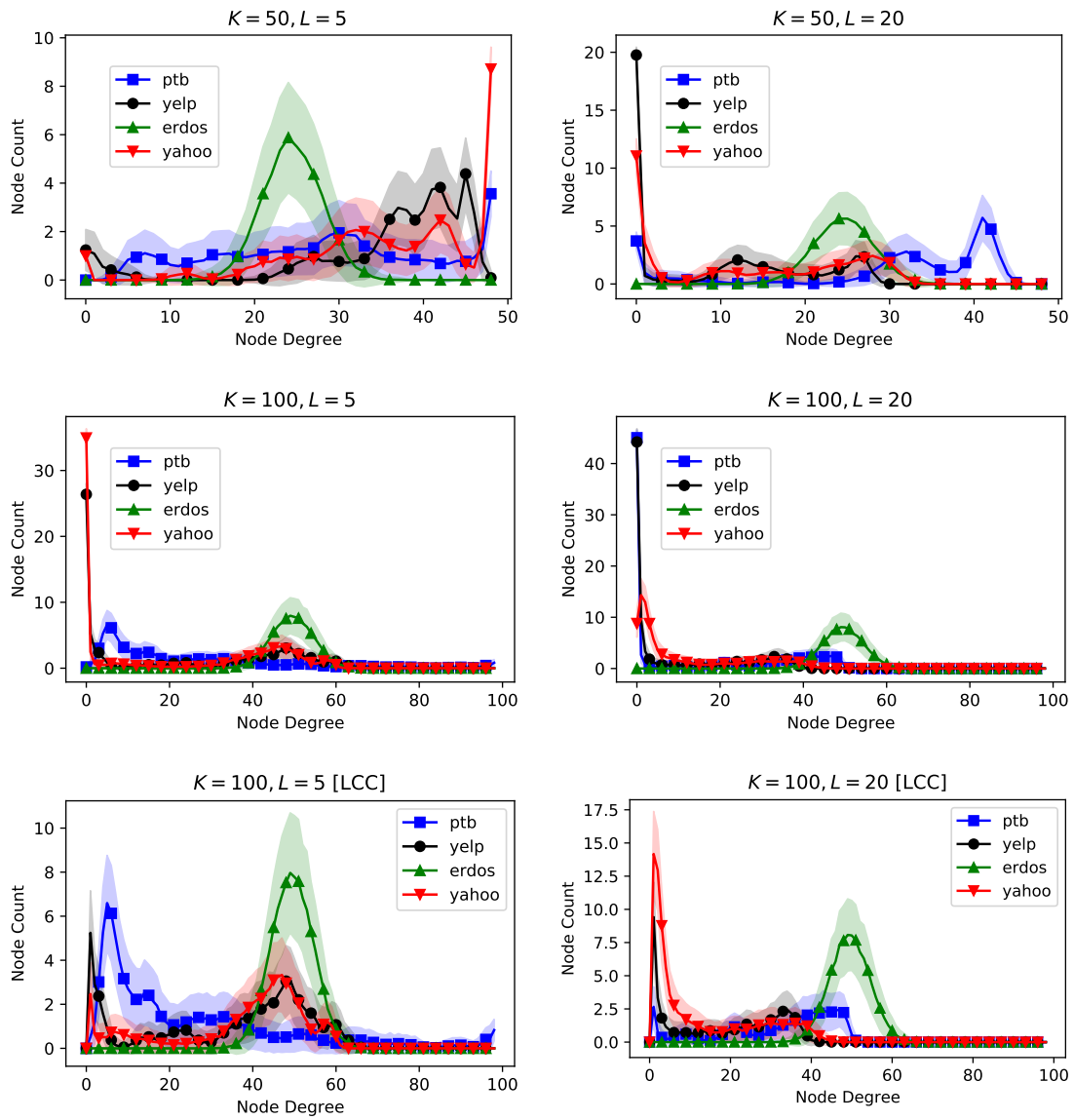
Figure 11: Degree distributions of inferred graphs from all corpora, compared to Erdos-Renyi graphs for $p = 0.5$. The upper four plots show results for full inferred graphs, the lowest two show the degree distributions of the largest connected component of the models for $K = 100$.

Interpolate Society & Culture Science & Mathematics

is steady eye contact good? when i am communicating with someone, i tend to give very steady, long _UNK eye contact. so i tried to _UNK it as a young naive girl, and now it's a habit i can't lose... it just depends on the person you are having the conversation with...

(1) i am 14 years old. their is this girl again who speaks very much of me and talks 2 me, the idea of me 2 her and never gives any suggestion to verbal _UNK for my 2nd. listen, therapy!

(2) what do you do when you think your best friend told you shes bisexual? when she says that, or you might have believed if your friend said it is. they're inevitably sharing that they don't share...

(3) how do you change liquid in an ice cube into liquid form? paste, mix and freeze _UNK a gallon of co2 into a _UNK and then _UNK in some ice to form a coating.

(4) what kind of rules does gravity apply? if a certain weight is placed in a container, the net force applied on it hits the water surface and the right weight will turn into gravity

how does a photovoltaic system that feeds back into the power grid get on the same phase angle? or? should i say does it need to be the same as the _UNK's?

Interpolate: Business & Finance Family – Relationships

at 35, am i too old to go to college to become a psychiatrist? i'm 37, and i just started my second semester in a 2 year college... you need to be prepare for the financial aspects, but the social ones are no problem...

(1) what would be a good title for a _UNK _UNK? i have _UNK in _UNK and there are no real courses done for it but i do love the job and i've already done my freshman year. i am currently teaching placement at _UNK and need the same as the average undergraduate student...

(2) has anyone here applied in the past 4 months or is it better to get a try out y _UNK a slightly better long term career

(3) lately im having trouble with my fiancee, how do i bring him back? it obvious at this point that you can't " bring us together ". try playing games.

(4) could i still go out with this guy and still be friends and respect him.? i don't want to just fell in love with the guy that i was with. i want 2 be with friend's girl and still be friends...

how do i know if my man, is inlove with me? well... some questions, how old are _UNK? - are you wealthy?, is he wealthy?, how long have you been together...

Table 9: Interpolation between four random instances from the Yahoo dataset

---

Interpolate: Very bad - bad

do not use this company!! they told me within one hour, then i called again they said the driver have 90 mins. 90 minutes later, they said the driver is in traffic and wait for 15 minutes, i checked google map no accident, all green on all freeway...

(1) i ordered for pick up as my daughter hadn't been told that or even ordered online. when i spoke to the young lady, who was _UNK, she carried on a conversation with not a manager. it's bad customer service and i wouldn't even bother with this place...

(2) place was clean... when i called to let them know i 'd get something else, the person that answering the phone wouldn't understand me... really? i gave this restaurant a b + for the cleanliness of the food and the friendliness of the staff

(3) i had the quesadilla and the carnitas tacos. i felt every bite of these were so rubbery and the potatoes were off. i feel like the service and the quality of food can do much better.

(4) somewhat disappointed. i did it once and loved it but today, today's water is bitter and salty... and the mint and cherry blossom _UNK'flavors just taste that way.

the food quality doesn't match the place at all. i think it's ok for a pub but this place is supposed to be a nice place for professional lunches. i had the chicken flatbread and the chicken was more like subway chicken! with so many options around that area i won't pick this place for lunch.

Interpolate: Very bad – Very Good

skip it... there are much better options out there! the " hot " food was not hot, and the flavor was only mediocre at most.

(1) indifferent to locals. the kids size pizzas were a billion times worse than a pizza hut. the quality of food was just awful. i wouldn't recommend this to a significant other for what it is.

(2) this new mexican spot is ok, bordering on childish. i went with friends and ordered a carne asada burro... it wasn't off the hook ; what made this place great were the chips & salsa sucked. yuck! ...

(3) wow. _UNK you give so much frosting!! we were a groupon special for a cupcake for the princess of chocolate, and we were pretty stoked. they were _UNK and creative. they even suggested we try the coconut ... we 'll definitely be back soon.

(4) went for the first time during a recent trip to vegas. our server jeff made special recommendations for our friends and i. it was fantastic most of the food was light and fresh...i would highly recommend this place!

i had dinner at republic kitchen tonight for the first time and was very impressed with the service, the decor, the menu, and the food quality... i am going back sunday for their brunch and jazz!

Table 10: Interpolation between four random instances from the Yelp dataset

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitation section

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

3, F

☑ B1. Did you cite the creators of artifacts you used?
1, 3, 4, 5, F

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
All datasets and artifacts we used are publicly available. Our code (attached as sup. material) shall also be make publicly available once the paper is accepted.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
The datasets we used are publicly available and widely used in research.

☒ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
The datasets we used are publicly available and widely used in research. Documentation about our own artifacts (models) can be found in the Appendix

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
E, F

C ☑ Did you run computational experiments?

4, 5, 6

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
E, F, I

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
E, F

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
E, F

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
F and source code

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.