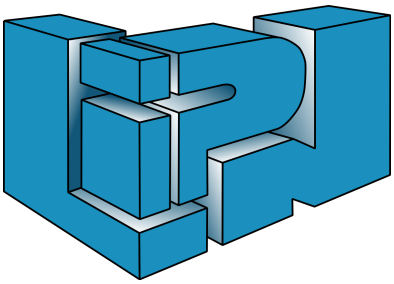
Représentations de phrases interprétables avec autoencodeurs variationnels et attention

Ghazi Felhi



Université Sorbonne Paris-Nord
Laboratoire d'Informatique de Paris-Nord

Interpretable Sentence Representation with Variational Autoencoders and Attention

Représentations de Phrases Interprétables avec Autoencodeurs Variationnels et Attention

Presentée par Ghazi Felhi

Soutenue, le 26 Janvier 2023, devant le jury composé de :

Mme. Adeline Nazarenko	Sorbonne Paris Cité	Directrice de thèse
M. Joseph Le Roux	Université Sorbonne Paris-Nord	Encadrant
M. Djamé Seddah	Université Paris Sorbonne	Co-encadrant
M. François Yvon	Université Paris-Saclay	Rapporteur
M. Benjamin Piwowarski	Sorbonne Université	Rapporteur
M. Laurent Besacier	Université Grenoble Alpes	Examinateur

Abstract

Recent progress in NLP can largely be attributed to progress in Deep Representation Learning techniques, which are based on opaque statistical learning methods. The success of these methods is due to the availability of large volumes of data which they leverage in pre-training schemes to improve generalization when fine-tuned on specific NLP tasks. Producing such high performance representations with an understandable meaning most often calls for the availability of annotated data. However, annotated data comes at great costs, and incurs recurring annotation effort which is impractical when deploying models at scale, i.e. for different languages, different tasks, and on different domains.

The purpose of this thesis is therefore to develop methods that enhance the interpretability of recent representation learning techniques, while accounting for the unavailability of annotated data. We choose to leverage Variational Autoencoders (VAEs), given their established efficiency in learning to relate observations to latent generative factors. VAEs have also been proven effective for semi-supervised learning, and interpretable representation learning, which makes them suitable for the research we tackle in this thesis.

As a first contribution, we identify two unnecessary components in the functioning scheme of Semi-Supervised VAEs, namely the Kullback-Leibler Divergence and the unobserved latent variable, and perform ablation experiments on them. Our experiments show that these components are indeed of no use to the semi-supervised VAE framework, and that removing them speeds-up computations. We also show that without these components, the model is easier to define and to train.

Our second contribution is based on VAEs, together with Transformers. Recent Deep Learning-based NLP is largely based on variants of the Transformer architecture, which is built using as a main component attention, a learning module which allows exchanging information between a set of elements in a parallel fashion. In previous works, attention has been shown to excel at spontaneously aligning structures from different languages, and to process language in a manner that exhibits patterns resembling understandable NLP concepts such as dependency trees. Based on such observations, we use Transformers attention to build two models with inductive bias to separate information in latent representations into understandable concepts without annotated data. This information separation in neural representations is a process called disentanglement.

The first model we present is an Attention-Driven VAE (ADVAE). It is the first VAE to use Transformers Cross-Attention to encode and decode vectorial latent variables. We experimentally demonstrate the ability of this model to separately represent, and separately control information about the realizations of core syntactic roles in sentences.

The second model we present, called QKVAE, is based on ADVAE and uses separate latent variables to form keys and values for the Transformer decoder it uses. The name QKVAE is a contraction of the Query, Key, Value (QKV) abstraction used in Transformers attention and the VAE acronym. We empirically demonstrate the ability of this model to separate syntactic information from semantic information in its neural representations. In experiments involving transfer of syntactic or semantic properties between sentences, QKVAE exhibits competitive performance when compared to previous supervised models, and equivalent performance to a previous supervised model that uses 50K annotated samples. Moreover, this final model displays improved syntactic role disentanglement capabilities compared to ADVAE.

In a context where text data is abundant but annotations are scarce, our work demonstrates that it is feasible to enhance the interpretability of state-of-the-art deep learning architectures for language modeling using only unannotated data.

Keywords: Variational Autoencoders, Transformers, Interpretability, Disentanglement, Semi-Supervised Learning, Unsupervised Learning, Language Modeling, Syntax.

Français

Titre : Représentations de Phrases Interprétables avec Autoencodeurs Variationnels et Attention

Résumé : Les progrès récents en Traitement Automatique de Langues (TAL) peuvent en grande partie être attribués aux progrès des techniques d'apprentissage de représentations profondes, qui reposent sur des méthodes d'apprentissage statistique opaques. Le succès de ces méthodes est dû à la disponibilité de grandes quantités de données utilisées en pré-entraînement afin d'améliorer leur généralisation lorsqu'elles sont adaptées (fine-tunées) à des tâches de TAL spécifiques. Produire de telles représentations qui soient à la fois performantes et compréhensibles nécessite souvent la disponibilité de données annotées. Or, ces données annotées sont très coûteuses, et doivent être collectées séparément pour les différents cas d'usage d'un modèle (i.e. langues, domaines, et tâches), ce qui n'est souvent pas envisageable pour des déploiements à grande échelle.

Le but de cette thèse est de développer des méthodes qui améliorent l'interprétabilité des techniques récentes d'apprentissage de représentation, tout en prenant en compte l'indisponibilité de données annotées. Nous avons choisi de nous appuyer sur les Auto-Encodeurs Variationnels (Variational Autoencoders ; VAE), compte tenu de leur efficacité établie dans l'apprentissage de la relation entre des observations et des facteurs génératifs latents. Il a également été montré que les VAE sont efficaces pour l'apprentissage semi-supervisé et l'apprentissage de représentations interprétables, ce qui les rend pertinents pour cette thèse.

Dans la première partie des contributions présentées dans cette thèse, nous identifions deux composants superflus dans le schéma de fonctionnement des VAE semi-supervisés, à savoir la divergence de Kullback-Leibler et la variable latente non observée, et nous effectuons des expériences d'ablation sur ces composants. Nos expériences montrent que ces composants n'ont effectivement pas d'utilité dans le cadre VAE semi-supervisé et que leur suppression accélère les modules utilisés. Nous montrons également que sans ces composants, le modèle devient plus facile à définir et à entraîner.

Notre seconde contribution repose sur les VAE, associés aux Transformers. Les techniques récentes de TAL qui reposent sur l'apprentissage profond s'appuient largement sur des variantes de l'architecture Transformer, laquelle est construite en utilisant comme

principale brique l'attention, un module d'apprentissage qui permet d'échanger des informations entre un ensemble d'éléments de manière parallèle. Des travaux précédents ont montré que l'attention excelle dans l'alignement spontané des structures de différentes langues et qu'elle présente des motifs ressemblant à des concepts linguistiques interprétables tels que les arbres de dépendance. Sur la base de ces observations, nous utilisons l'attention des Transformers pour construire deux modèles dont le biais inductif permet de séparer l'information dans les représentations latentes en concepts compréhensibles sans données annotées. Cette séparation d'information dans les représentations neuronales est un processus appelé désenchevêtrement (disentanglement).

Le premier modèle que nous présentons est un VAE à attention (Attention-Driven VAE ; ADVAE). C'est le premier VAE à utiliser l'attention croisée des Transformers pour encoder et décoder des variables latentes vectorielles. Nous démontrons expérimentalement la capacité de ce modèle à représenter séparément les informations sur les réalisations de rôles syntaxiques essentiels (core syntactic roles) dans les phrases et à les contrôler.

Le second modèle que nous présentons, appelé QKVAE, dérive de ADVAE et utilise des variables latentes séparées pour formuler des clés et des valeurs pour le décodeur Transformer qu'il utilise. Le nom QKVAE est une contraction du triplet Query (Requête), Key (Clé), Value (Valeur) utilisé dans l'attention des Transformers et de l'acronyme VAE. Nous démontrons empiriquement la capacité de ce modèle à séparer les informations syntaxiques des informations sémantiques dans ses représentations neuronales. Dans des expériences de transfert de propriétés syntaxiques ou sémantiques entre les phrases, QKVAE présente une performance compétitive par rapport aux précédents modèles supervisés et une performance équivalente à un précédent modèle supervisé utilisant 50 000 échantillons annotés. De plus, ce modèle final présente des capacités de désenchevêtrement de rôles syntaxiques améliorées par rapport à ADVAE.

Dans le contexte actuel où les données textuelles sont abondantes et où les données annotées sont difficiles à obtenir, notre travail démontre qu'il est possible d'améliorer l'interprétabilité des architectures de deep learning de pointe pour les modèles de langue à partir de données non annotées.

Mots-clés : Autoencodeurs Variationnels, Transformers, Interpretabilité, Désenchevêtrement, Apprentissage Semi-Supervisé, Apprentissage Non-Supervisé, Modèle de Langue, Syntaxe.

Acknowledgements

I would like to express my sincere gratitude to my thesis jury members, Laurent Besacier, Benjamin Piwowarski and François Yvon for their meticulous review of my thesis and insightful comments.

I feel deeply grateful for having had my supervisor Joseph Le Roux as the person with whom I worked most closely. His deep insights into my work, his extensive knowledge, and his unwavering positivity were crucial to both my work *and* well-being. I also wish to thank my supervisor Djamé Seddah for helping me improve on various regards, for providing invaluable links and references to parts of the linguistics literature I could not navigate, and for bearing with my often unbearable stubbornness. I also have my Ph.D. director Adeline Nazarenko to thank for her scientific, emotional, and even administrative guidance throughout the 3 past years.

I am thankful to my colleagues at LIPN to whom I owe a lot of good memories, and with whom I wish I spent more time than COVID allowed us to. I thank Victor for not getting tired of me asking questions about physics, Alex for teaching me lots of cool ice skating tricks, and both of them plus Francesco for indulging in our 3 hour One Piece-centered debates every Friday Morning. I thank Dasha for teaching me a dance I still can't name and letting me practice my mediocre Russian on her, and Théo for taking interest in my Trello system (I hope you still use it) and for showing me that french people can in fact pronounce ق . I would also like to thank Mathieu for doing the most random things every time I looked his way, and Will for teaching me that Australian spiders were actually cool and that the real danger was snakes.

To my friends who collectively managed to make 2 years of pandemic as pleasant as could be, thank you. Namely, Léo, who first made me want to pursue a Ph.D., Firas who embarked on a similar journey and helped me through the difficult times by simply saying he was going through the same things, Dali, my roommate, who bore with my unwashed dishes and my chaotic sleeping schedule, Sami who's had the life sucked out of him by his startup but still found time to be a good friend, and numerous other people who oftentimes broke the curfew to come and help me get more noise complaints from my neighbors.

Finally, I extend my heartfelt appreciation to my family to whom I owe my birth, this

work, and everything in between. I would like to express my gratitude to my two sisters, Hiba and Rym, whom I was proud to introduce to my friends after the presentation, and my parents Souad and Miled who came all the way from Tunisia to see me present my work in Paris despite the horrid travel logistics.

Contents

List of Tables

List of Figures

Part I

Preamble

Chapter 1

Introduction

Natural Language Processing (NLP) has come a long way in the recent years due to a paradigm shift towards Deep Learning-based systems. A major catalyst for the advances it has seen, is an intense focus on Representation Learning for language, *i.e.* designing embedding techniques for language (*e.g.* word embeddings or sentence embeddings). These embeddings are the cornerstone to all state-of-the-art NLP systems, since mapping textual observations to numerical vectors is the starting point of all these systems. Consequently, over the past years, better representation learning for language has consistently translated to better performance on downstream tasks such as parsing [Chen and Manning, 2014], translation [Devlin et al., 2014], or natural language inference [Bowman et al., 2015]. Over the last decade, representation learning in NLP evolved from static vectors for context-agnostic word-level representation [Mikolov et al., 2013], to contextual, and thus dynamic, subword-level representations [McCann et al., 2017].

An essential ingredient to the rapid growth of the literature on contextual language representation is the Transformer architecture [Vaswani et al., 2017]. This architecture has been built to provide sequence elements (tokens) with context information while solely relying on a mechanism called attention. Contrary to the serial processing occurring in Recurrent Neural Networks (RNNs), attention allows sharing information between sequence elements in a completely parallel fashion, which allows faster learning and inference. The first Transformer-based language representation model, BERT [Devlin et al., 2019], led to a significant improvement on language understanding benchmarks and paved the way for a surge of similar models [Yang et al., 2019, Lan et al., 2020, Liu et al., 2020], later dubbed BERT-like models.

The race to performance caused representation learning models to grow larger, deeper, and thus less and less interpretable. In response to this, great efforts have been deployed to read into the inner workings of these models [Jawahar et al., 2019a, Hu et al., 2020, Kodner and Gupta, 2020, Marvin and Linzen, 2020, Kulmizev et al., 2020, Rogers et al., 2020a], and procedures aimed at grounding language representation into understandable concepts have grown to become a major research direction in the NLP community. Here again, attention, the cornerstone of BERT-like models, was essential to a great number of

insights into state-of-the-art NLP models. As a matter of fact, a byproduct of its design is that it calculates values that determine how much context information is pulled from a token to another, which provides a reading into the interaction between these tokens. For instance, Clark et al. [2019] have shown that attention in BERT spontaneously specializes in dependency parsing at different parts of the network, an observation that was the basis for the current state-of-the-art unsupervised dependency parsing system [Shen et al., 2022].

As an alternative to *post-hoc* analyses on black-box models, interpretability in NLP models can be tackled by building models which are understandable by design. This built-in interpretability can be pursued at different levels in an NLP system. For instance, there have been efforts to plug knowledge graphs into language models [Logan et al., 2019, Peters et al., 2019] so as to freely *control* the knowledge used by these models for generation. Other types of interpretability-oriented interventions on NLP models include *interpretable* inference schemes for multi-hop question answering where, for example, Weber et al. [2019] integrate a Prolog prover into their system and Saha et al. [2019] turn questions into series of programmatical queries. In essence, built-in interpretability efforts are channelled into solving two research questions:

- How can we obtain *interpretable* inner-representations in neural models ? *(i.e. interpretable encoding)*
- How can we control *interpretable* aspects of the output of neural models ? *(i.e. interpretable decoding)*

To tackle the above problems and design a system that is interpretable with regard to a certain aspect, the classical procedure is to collect samples annotated with this aspect. Subsequently, one correlates these annotations to some inner-representations in the system, or uses them as a conditioning factor for the outputs of the system. For example, provided text samples annotated with their style, one could train neural representations to separately encode information that correlates with style, modify it, then decode it to obtain a sentence with similar meaning but different style [Wang et al., 2019]. However, this procedure assumes access to annotated samples, which are often costly, especially when the annotation procedure requires expertise. For instance, [Seddah et al., 2020] report an annotation cost of 87k€ for 1500 sentences to be fully annotated in morpho-syntax and Universal Dependency[Nivre et al., 2016] syntax. The cost of labeled data is not only high, but also recurrent since adapting to data that continuously evolves such as language or to new domains[1] entails a fine-tuning of the learned models, and therefore a new round of annotation. Given the considerable volumes of data needed for training neural models and the plurality of languages and domains, there is clearly little hope for classical supervised learning to provide coverage with interpretable language technologies to a sufficient proportion of their possible use cases.

[1]For reviews on concept drift and domain adaptation, readers may refer, respectively, to Lu et al. [2019c] and Farahani et al. [2021]

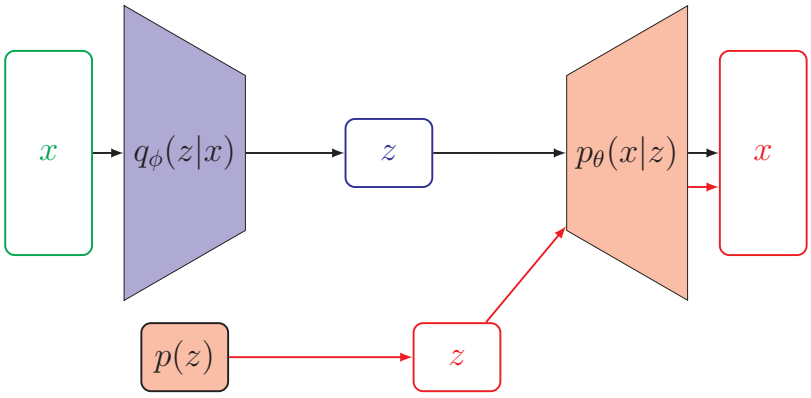


Figure 1.1: A diagram sketching the functioning scheme of a Variational Autoencoder. x is an observation, $q_\phi(z|x)$ is an encoder, $p_\theta(x|z)$ is a decoder, and $p(z)$ is a prior.

To summarize the above, there is a dire need for methods that would improve interpretability for neural NLP models while requiring little-to-no annotated text samples. Accordingly, we turn to Latent Variable Models (LVMs), a framework where it is possible to relate observations to a carefully crafted latent structure of probabilistic variables. This framework provides a principled means to inject beliefs in models through the choice of distributions (*e.g.* Gaussian, Categorical, etc) and latent structures (*e.g.* trees, lattices, independent variables) which can be leveraged to counteract the lack of annotations.

The flagship of LVMs in the Deep Learning era is the Variational Autoencoder (VAE; Kingma and Welling, 2014). In that sense, the purpose of this thesis is to explore and improve on the current usage of VAEs in NLP, so as to ease their application, and to create neural representations of language with a clear grounding to understandable linguistic factors, while requiring little-to-no supervised learning.

1.1 VAEs for Explainable NLP

We give, in this section, a quick overview as to what constitutes a VAE. We also give a broad description of its usages in NLP in general, and how it proved useful for explainable NLP in particular. The brief explanations given here are elaborated upon in a dedicated background chapter (Chapter 3).

1.1.1 What is a VAE ?

The components forming a VAE are depicted in Figure 1.1. Given a set of observations x, VAEs are a class of Deep Learning models that train a generative model $p_\theta(x) = \int_z p_\theta(x|z)p(z)dz$, where $p(z)$ is a prior distribution on latent variables z that serve as a seed for generation, and $p_\theta(x|z)$, called the decoder, generates an observation x from each latent variable value z.

Since directly maximizing the likelihood $p_\theta(x)$ to train a generative model is rarely tractable[2], an approximate inference distribution $q_\phi(z|x)$, called the encoder, is used to

[2]Maximizing this likelihood is possible for cases where the support of z is small enough for summation. However latent variables with such a small support are, in general, not expressive enough to describe the distribution of

formulate a lower-bound to the exact log-likelihood of the model, called the Evidence Lower-Bound (ELBo). This lower-bound is formulated as follows:

$$\log p_\theta(w) \quad \geq \quad \mathbb{E}_{(z)\sim q_\phi(z|w)}\left[\log p_\theta(w|z)\right] \; - \; \mathrm{KL}[q_\phi(z|w)||p(z)] \quad = \quad \mathrm{ELBo}(w;z) \quad (1.1)$$

where KL is the Kullback-Leibler divergence. Although this objective still requires calculating an expectation over z, in practice, it can be approximated with sampling-based estimates and used to efficiently learn a generative model.

The VAE framework combines the efficiency of Variational Inference [Jordan et al., 1999] with the representational power of Deep Learning. Its generative capabilities, as well as its encoder-decoder architecture made it into a Swiss Army knife for NLP, as is explained in the next section.

1.1.2 An Overview of VAEs in NLP

The first work in NLP exploring the use of VAEs was Bowman et al. [2015]'s work on language generation from smooth continuous representations, where these continuous representations are VAE latent variables. The smoothness mentioned by Bowman was a byproduct of the fact that VAE encoders are probability distributions, as opposed to the deterministic scalars yielded by classical Sequence Autoencoders [Sutskever et al., 2014]. As a matter of fact, the regularization provided by the sampling procedures occurring during training produces *smooth* vicinities in the VAE latent space where intervening on representations doesn't damage the well-formedness of output sentences. Moreover, many recent works have shown that VAEs have a natural tendency toward producing disentangled representations with their encoders [Higgins et al., 2017, Rolinek et al., 2019], *i.e.* representations where understandable concepts are localized in identified neurons. This ability to disentangle, together with the smoothness property, allowed various works on interpretability to produce understandable representations with VAE encoders, and to intervene on these representations and decode them to obtain well-formed samples where they successfully modify the factors they explicited in their representations [Chen et al., 2019a, Cheng et al., 2020, Huang and Chang, 2021b]. Although these works produced encouraging results, they largely assume the availability of annotated data and therefore mostly rely on supervised learning. In contrast to these works, the present thesis focuses on data-efficient methods to minimize the need for annotated data.

After the introduction of VAEs, Kingma et al. [2014] also showed that they could be used for semi-supervised learning, where a classifier is trained normally on annotated samples, and as a VAE encoder on non-annotated samples. This use of VAEs also propagated to NLP improving data efficiency across several tasks [Wolf-Sonkin et al., 2018, Habib et al., 2019, Corro and Titov, 2019, Chen et al., 2018a]. Semi-Supervised Learning with a VAE produces an Autoencoder where the latent representation is understandable while requiring relatively few annotated samples. It is therefore an important avenue for re-

target observations x.

search on both interpretable representations learning (with the encoder), and controllable generation (with the decoder).

1.2 An Outline of this Thesis

Posterior to the present introductory part of this thesis, we dedicate a part to background notions which are necessary to understand our contributions, we present our semi-supervised learning-specific contribution in a third part, then our disentanglement-specific contribution in a fourth part. A fifth and final part concludes our thesis with a summary of our findings and a few perspective research directions.

Part II: Background This part details all the notions necessary to understanding the contributions presented in this thesis. First, in Chapter 2, we go over neural language modeling, as it is an essential brick to generative NLP systems, and the basis to VAE language models which are what we aim to build in an interpretable way throughout the thesis. The chapter first provides explanations on how to build a standard Language Model (LM), how to build a conditional LM (to lay the groundwork for the introduction of latent variables), the techniques to sample from LMs, and the most common evaluation protocols for these models. Masked Language Models are also be explained as they are an important component of recent representation learning techniques.

Chapter 3 is about VAEs, since they are the machine learning framework used throughout all contributions in this thesis. It covers its theoretical foundations (*i.e.* architecture and loss function), the implementation details necessary for it to work, and how importance weighting is used to approximate its perplexity, the main evaluation metric for language modeling. It also discusses posterior collapse, an issue that is specific to language modeling with VAEs, and the different solutions that were proposed to deal with it in the literature. Finally, we discuss in this chapter the way VAEs were adapted for semi-supervised learning.

Chapter 4 is about a core notion for this thesis : Disentanglement, *i.e.* the process of separating understandable concepts in neural representations. The chapter justifies the need for disentanglement and explains how it works. It also provides technical justifications for the use of VAEs for disentanglement, and gives precise reasons as to why and when unsupervised disentanglement works with VAEs. This chapter also describes the evaluation protocols used to measure disentanglement, and previous works in NLP pertaining to this area of research.

Chapter 5 revolves around Transformers, the architecture used to implement the inductive bias used for our disentanglement-specific contributions. We first discuss in this chapter attention, the core component of Transformers, then the way the Transformer architecture is built with different uses for attention and other carefully crafted components. The remainder of this chapter consists in a summary of Transformer-related works throughout NLP, namely (causal and masked) language modeling with Transformers, and Transformer-specific model analyses.

The final background chapter, Chapter 6, goes over a few notions regarding syntactic analysis that we leverage to explain our disentanglement-specific contributions. Specifically, this chapter summarizes Context-Free Grammars and how they produce constituency trees and presents dependency parsing and the notion of syntactic role. The explanations on dependency analysis also focus on the distinction between oblique and core syntactic roles and how the latter relate to the predicate-argument structure.

Part III: Semi-Supervised Learning with VAEs As a first contribution, we chose to study the use of VAEs in semi-supervised learning (Chapter 7). In this context, textual observations are described with 2 latent variables: *i)* a partially-observed latent variable with a known meaning for which we have a small amount of annotations, *ii)* an unobserved latent variable that describes factors other than that of the partially-observed variable, and for which we have no annotated data. Semi-Supervised VAEs (SSVAE) can be used to leverage unlabeled data to improve the inference of a factor for which we only have a small amount of annotations. In this chapter, we provide a detailed description of the source of this improvement, and use it to ablate two sources of over-complexity in the SSVAE framework. These ablations yield a model that is smaller, faster, and easier to define, while preserving the performance of the original framework.

Part IV: Unsupervised Disentanglement of Sentence Representations The bulk of our contributions pertains to disentanglement of sentence representations where we build, with Transformers attention, latent variable models with inductive bias that are capable of giving rise to understandable concepts in the latent representations without annotations *i.e.* in an unsupervised fashion. These contributions are presented in this part over two chapters.

In chapter 8, we tackle unsupervised learning of sentence representations that display separation (disentanglement) with regard to the realizations of their *core* syntactic roles. Drawing from the observation that attention-based Neural Machine Translation systems are capable of aligning spans between languages in a coherent fashion, we introduce an Attention-Driven Variational Autoencoder (ADVAE). ADVAE is the first VAE that uses Cross-Attention to encode and decode latent variables. ADVAE also enables using attention to quantify the interaction of latent variables with inputs. The assessment of this model required designing an evaluation procedure that enables quantifying the disentanglement of realizations of syntactic roles, both in the encoder and in the decoder of our model. Our experimental results show that it is indeed possible with ADVAE to obtain sentence representations where the realizations of syntactic roles exhibit pronounced separation without using supervised learning. We also show that our model is capable of separately changing the realizations of core syntactic roles in sentences it generates, and that it does it better than classical LSTM-based or Transformer-based text VAEs. However, we show that the success of this separation is limited to the case where a dataset consists of regularly structured sentences.

In the same line of work, we tackle in Chapter 9 disentanglement of structure from content in sentence embeddings. By leveraging the internal variables of the cross-attention mechanism in the decoder of our previously built ADVAE, we define a variable that is enforced to only control the *keys* in the decoder's Cross-Attention, and control the *values* using the remaining latent variable. We experimentally demonstrate the ability of the key-specific latent variable to channel syntactic information, while leaving semantic information to the value-specific latent variable. In experiments where the resulting model, QKVAE, is set to transfer syntax and semantics, we show that it performs competitively compared to supervised counterparts, and that a previous supervised model needs more than 50K annotated samples to outperform it. Experiments measuring its ability to disentangle realizations of syntactic roles also show that it improves upon ADVAE, and that it mitigates its inability to disentangle information from different syntactic roles on datasets where sentences do not display regular structure.

Part IV: Conclusion and Perspectives This part concludes this thesis by summarizing our findings, and describing a few research directions which may be pursued on the basis of the contributions presented in this thesis. Specifically, it elaborates the way a structured latent variable model, as opposed to the independent latent variable models described in this thesis, can alleviate some of the present shortcomings of our last model, QKVAE. As a last note, we emphasize the ease of applicability of our models to non-text data, since they are unsupervised and modality-agnostic, and thus argue their potential for producing explainable representations for other modalities (*e.g.* images) or in the multi-modal setup.

1.3 Publications Related to this Thesis

- Ghazi Felhi, Joseph Le Roux, and Djamé Seddah. 2021. Challenging the Semi-Supervised VAE Framework for Text Classification. In Proceedings of the Second Workshop on Insights from Negative Results in NLP, pages 136–143, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. [Felhi et al., 2021a] **(Chapter 7)**

- Ghazi Felhi, Joseph Le Roux, and Djamé Seddah. 2021. Towards Unsupervised Content Disentanglement in Sentence Representations via Syntactic Roles. Presented at CtrlGen: Controllable Generative Modeling in Language and Vision. Online, co-located with NeurIPS 2021. [Felhi et al., 2021b] **(Chapter 8)**

- Ghazi Felhi, Joseph Roux, and Djamé Seddah. 2022. Exploiting Inductive Bias in Transformers for Unsupervised Disentanglement of Syntax and Semantics with VAEs. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5763–5776, Seattle, United States. Association for Computational Linguistics. [Felhi et al., 2022] **(Chapter 9)**

Part II

Background

Chapter 2

Neural Language Modeling

Language Modeling is a classical NLP task where one builds generative models for a language. These models can be learned using text corpora and functions with learnable parameters that allow assigning probabilities to text samples quantifying how likely these samples are to belong to these corpora. Tractably assigning probabilities to text samples implies that these models enable *sampling* text samples, *i.e. generating* samples which are likely to come from the training data distribution.

This resulting ability to generate text samples is essential to most *generative* (as opposed to *discriminative*) tasks in NLP: namely the design of Dialogue Systems [Zhang et al., 2018], Machine Translation [Bahdanau et al., 2015], Textual Style Transfer [Li et al., 2018], Abstractive Summarization [Chopra et al., 2016], *inter alia*.

Besides the utility of Language Models (LMs) as a brick for the above NLP systems, Radford et al. [2019] have shown through their the large-scale training of an LM, GPT-2[1], that modeling the distribution of text corpora requires learning a wide set of skills such as Reading Comprehension, Question Answering, and even Machine Translation. Subsequent large-scale LMs such as GPT-3 [Brown et al., 2020] have also shown that LMs can exhibit proficiency at advanced reasoning skills such as few-digit arithmetics and fake news generation. This stresses the centrality of LMs to NLP in particular, and to AI in general.

In this thesis, we aim to build *interpretable* Neural LMs. To lay the groundwork for our contributions, we explain the intuitions behind LMs, and formally define the learning modules and objectives used to obtain them (§ 2.1). The particular class of LMs that we base our work on are Variational Auto-Encoder-based LMs which condition LMs on latent variables. In that regard, we introduce Conditional LMs (§ 2.2) that are later used to plug latent variables into LMs in Chapter 3. Next, we introduce sampling techniques we use to obtain output text samples from LMs(§ 2.3), and metrics that are usually applied to quantify the quality of an LM(§ 2.4). Finally, we explain MLMs(§ 2.5), which are the basis to recent Transformer-based representation learning techniques.

[1]The GPT-2 experiment is explained in more detail in § 5.3.2

2.1 Learning Language Models

Consider a corpus U[2] of text samples w. An LM, learned on the basis of this corpus, consists in a probability distribution p_θ, where θ is a set of parameters, that should assign high probabilities to samples that belong to U, and low probabilities to samples that are unlikely to come from U. To represent a text sample w, we break it down to a series of tokens $w_i \in V$ where V is a predefined vocabulary (*e.g.* words, characters, sub-word units, etc). Using these tokens the probability of a sample is then measured as $p_\theta(w) = \prod_i p_\theta(w_i|w_{<i})$, *i.e.* one token w_i at a time conditioned on the tokens $w_{<i}$ that precede w_i in w. This model is referred to as an *autoregressive* or *causal* LM[3], and the default learning strategy to estimate θ is simply to maximize the probability (or log-probability) of samples in U according to p_θ as follows[4]:

$$\underset{\theta}{\operatorname{argmax}} \sum_{w \in U} \log p_\theta(w) = \underset{\theta}{\operatorname{argmax}} \sum_{w \in U} \sum_{i}^{|w|} \log p_\theta(w_i|w_{<i}) \tag{2.1}$$

The above learning strategy is called Maximum Likelihood Estimation (MLE)[5]. Given this strategy, all that remains is to define $p_\theta(w_i|w_{<i})$.

LMs can be approached in a naïve way by restricting the dependency of the current word's probability estimation to the $n-1$ previous words, *i.e* $p_\theta(w_i|w_{<i}) = p_\theta(w_i|w_{i-(n-1)}, \ldots, w_{i-1})$. This approach, called n-gram language modeling, is limited in that it can only account for a *fixed* number of previous words but it is fairly simple to implement.

Among other design choices, n-gram LMs can be tackled with a linear model as follows: Let E and C respectively a word embedding matrix and a context embedding matrix that store for each word w_i in the vocabulary V representations $E_{w_i} \in \mathbb{R}^{D_E}$, and $C_{w_i} \in \mathbb{R}^{D_C}$. Using a linear transformation defined by the matrix $M \in \mathcal{M}(nD_C, D_E)$, and the concatenation operator Cat one can define an n-gram LM as follows:

$$p_\theta(w_i|w_{<i}) = p_\theta(w_i|w_{i-(n-1)}, \ldots, w_{i-1}) = \text{softmax}_{w_i}(s) \tag{2.2}$$

$$\text{s.t. :} \quad \text{softmax}_{w_i}(s) = \frac{exp(s_{w_i})}{\sum_{s_{w_j} \in s} exp(s_{w_j})} \tag{2.3}$$

$$s = \{s_{w_j};\ \text{s.t. } s_{w_j} = E_{w_j} M \hat{C}, w_j \in V) \tag{2.4}$$

$$\hat{C} = \text{Cat}(C_{w_{i-(n-1)}}, \ldots, C_{w_{i-1}}) \tag{2.5}$$

[2]The notation U refers to the fact that samples are *unlabeled*, *i.e.* not paired with annotations on some explaining factor for each observation.

[3]Non-autoregressive language modeling is outside of the scope of this thesis. Despite being less common, it is an active area of research [Li et al., 2022],

[4]The objective described in Equation 2.1 may be accompanied by a regularization term on the parameters θ which is added to the objective as an auxiliary term $R(\theta)$.

[5]Alternative learning strategies have been proposed in the literature (*e.g.* Sequential Generative Adversarial Networks; Yu et al., 2017), but MLE remains dominant for language modeling.

The above procedure can be summarized in the following steps:

- Calculate a representation $\hat{C}$ for the context or the word w_i.
- Score all word in the vocabulary with regard to the current context using a score function s.
- Calculate a normalized version of the score s that will be used to *parameterize* the categorical distribution $p_\theta(w_i|w_{<i})$.

This procedure is the standard procedure used to define an LM even beyond n-gram language modeling. LMs have been improved across various stages of this procedure compared to the above naïve model. For instance, the work of Yang et al. [2018] has shown that the softmax operator can only model as many contexts as the number of dimensions used for its input matrix, and therefore represents a bottleneck for the scoring of contexts with regard to words. However, research on LMs is most active on the context representation $\hat{C}$. As a matter of fact, a version of the above n-gram LM has been explored by Bengio et al. [2000] with neural networks to calculate word representations and achieved, at the time, state of the art language modeling performance with $n = 5$.

In the early 2010's, LMs started incorporating Recurrent Neural Networks (RNNs) to represent context over indefinitely long series of previous words [Mikolov et al., 2010]. These neural modules enable estimating a fixed size representation h_N for context windows of arbitrary length N using the following recursive rule:

$$a_N = b + Wh_{N-1} + UC_{w_N} \tag{2.6}$$

$$h_N = \tanh(a_N) \tag{2.7}$$

where b, W and U are learnable parameters, and tanh is the hyperbolic tangeant function. In a similar spirit, previous works have shown that state-of-the-art language modeling performance can be achieved using variants of RNNs, for instance LSTM [Merity et al., 2018] or Mogrifier LSTM [Melis et al., 2020]), and neural models thus became the *de facto* approach to parameterize LMs.

2.2 Conditional Language Models

In contrast to vanilla LMs, controllable or conditional LMs require an input constraint on p_θ that steers the probability estimation towards a desired property. A simple example can be that of an LM trained on movie reviews, where we try to *control* the sentiment (positive or negative) of the generated reviews. Another example, that is rarely referred to as a conditional LM, is that of Machine Translation where we aim to generate text samples in a certain language conditioned on their counterpart in another language.

To model controllable language generation, one simply *conditions* the output probability distribution on an additional factor y such that word probabilities at each generation step

become parameterized by $p_\theta(w_i|w_{<i}, y)$. Together with the previous words $w_{<i}$, y is used to estimate a context representation $\hat{C}$ that will be trained to assign high probabilities to words that realize text samples with the desired underlying attribute. Given a labeled dataset L consisiting in text samples w coupled with annotations y on the attribute that one aims to account for, learning a conditional LM is straightforward using a slightly modified version of the Maximum Likelihood objective introduced in 2.1:

$$\underset{\theta}{\operatorname{argmax}} \sum_{w \in U} \log p_\theta(w|y) = \underset{\theta}{\operatorname{argmax}} \sum_{(w,y) \in L} \sum_{i}^{|w|} \log p_\theta(w_i|w_{<i}, y) \tag{2.8}$$

However, when the underlying factor we aim to model is only partially-observed in the data at hand, or when it is never observed[6], the above objective must be swapped for another that accounts for the data points that only consist of an unlabeled observation[7] w. An objective that enables working with this constraint is described later in this thesis in the context of semi-supervised learning with VAEs (§ 3.7).

2.3 Sampling Techniques for Autoregressive Language Models

The canonical sampling procedure to generate text from LMs, random sampling, is to pick words according to their probabilities $p_\theta(w_i|w_{<i})$ until a maximum length is reached, or an end-of-sequence flag is triggered. LMs are typically trained on text samples bracketed by a beginning of sentence token $<$ bos $>$ that plays the role of $w_{<i}$ for the first generation step, and an end of sentence token $<$ eos $>$ that signals the end of the text sample, and thus the token sampling loop.

Random sampling makes sense when the model needs to generate diverse samples. An instance where this can be useful is data augmentation where the model needs to augment a dataset with diverse samples in order to improve the generalization of methods learned on this dataset [Han et al., 2019]. In contrast to such case, many applications such as Machine Translation, require picking *the most likely* sequence of tokens rather than *a likely* sequence of tokens. In this case, generation is most often referred to as *decoding*[8] since it is required to roll out an expected response within a narrow range of possibilities based on precise contextual clues.

Finding the most likely sequence of tokens requires estimating the probabilities of all possible sequences, which is intractable. Therefore, decoding is usually tackled using heuristics, where the 2 most commonly used heuristics are Greedy Sampling and Beam

[6]Later in this thesis, contrary to observed or partially-observed latent variables that we denote y, we denote unobserved latent variables z.

[7]For example, one could marginalize the objective described in 2.8 over y, if this marginalization is tractable (which is rarely the case).

[8]This most frequently used term, *decoding*, is short for *Maximum A Posteriori (MAP)* decoding, where one tries to produce the sequence with highest overall probability. Other decoding strategies, such as *minimum Bayes risk decoding* [Kumar and Byrne, 2004] may be encountered in the literature [Eikema and Aziz, 2020] but the MAP decoding is by far the dominant paradigm.

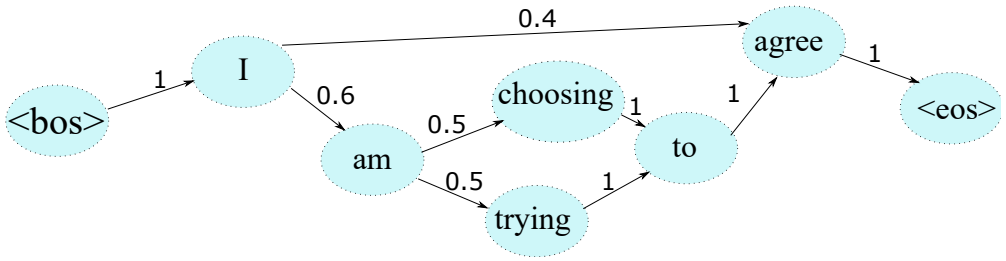


Figure 2.1: Example transition probabilities for a decoding process.< bos > and < eos > are respectively beginning-of-sentence and end-of sentence tokens. Transition probabilities are transcribed over arrows linking different tokens. The absence of an arrow means a null transition probability.

Search. Greedy sampling is fairly straightforward: one simply picks, at each generation step, the word with the highest probability. However, picking the most likely word at each generation step does not guarantee sampling the highest-probability sequence. An example of how this can occur is illustrated in Figure 2.1. When only the token I has been picked, $p_\theta(am|I) > p_\theta(agree|I)$. However, the probability of the sentence that follows from choosing *agree* is:

$$\begin{aligned} & p_\theta(I, agree, < eos > | < bos >) = \\ & p_\theta(I| < bos >) \quad p_\theta(agree| < bos >, I) \quad p_\theta(< eos > | < bos >, I, agree) \\ =1* & \quad 0.4* \quad 1 \quad = 0.4 \end{aligned} \tag{2.9}$$

On the other hand, the probability of a sentence that follows the choice am[9] is:

$$\begin{aligned} p_\theta(I, am, choosing, to, agree, < eos > | < bos >) = & \\ & p_\theta(I| < bos >) p_\theta(am| < bos >, I) \\ & p_\theta(choosing| < bos >, I, am) p_\theta(to| < bos >, I, am, choosing) \\ & p_\theta(agree| < bos >, I, am, choosing, to) \\ & p_\theta(< eos > | < bos >, I, am, choosing, to, agree) \\ & = 1 * 0.6 * 0.5 * 1 * 1 * 1 = 0.3 \end{aligned} \tag{2.10}$$

Therefore, although picking the best option at each generation step favors the token am at the second step, the overall highest probability sentence is rather obtained by picking *agree*.

To better explore word transition graphs such as the one represented in Figure 2.1, we use Beam Search. This method is initialized by picking the k highest-probability tokens to initialize k sequences, where k is called the beam width. In what follows, Beam Search measures for the k sequences the probability for each token in the vocabulary V to be the next token. The combination of each sequence with each word from the vocabulary yields an array of sequences of size $k * |V|$ from which k highest-probability *sequences* will be

[9] Branching on *choosing* or *trying* leads to the same sentence probability here.

selected. The process is repeated until all k sequences have reached the $<$ eos $>$ flag or a predetermined maximum length. Here we emphasize that, contrary to Greedy Decoding, Beam Search keeps or drops sequences on the basis of the entire sequence probability (*i.e.* $p_\theta(w_i, w_{i-1}, \ldots, w_1)$) instead of the next token probability (*i.e.* $p_\theta(w_i|w_{i-1}, \ldots, w_1)$). While guaranteeing equal or higher-probability decoded sentences, Beam Search requires using k times the memory required for Greedy Sampling where typical values for k range from 5 to 20.

Next to Beam Search and Greedy decoding, other decoding schemes have been developed in order to better imitate human language. Specifically, Holtzman et al. [2020] show that aiming for high probability sentences through Beam Search and Greedy Decoding, even from high quality LMs, leads to sentences that have low diversity compared to human-generated text. On the other hand, unhinged random sampling results in erratic and often unnatural sequences. A middle-ground can be reached using methods like Top-k Sampling, or Holtzman et al. [2020]'s Nucleus Sampling which truncate the range of possible tokens for random sampling to a restricted high-probability list. Although the output of these methods may exhibit more human-like patterns, we restrict our study to outputs from Greedy Sampling or Beam Search, which are designed with the intent[10] to produce maximally likely sequences, and are therefore better suited for inquiries about what the model has learned.

2.4 Evaluation the Generative Capabilities of Language Models

Similar to sampling techniques, evaluation strategies for generative models depend on whether the model is required to describe a range of outputs with its samples, or to produce a specific output given some contextual clues like previous words used as a prompt, or extrinsic factors encoded in the context representation.

In the case where no specific output is required from the LM, we aim to assess the well-formedness[11] of its outputs. Ideally, an oracle would look at samples from the LM, and score their syntactic and semantic soundness. But since such an oracle is rarely available, the norm is to evaluate the model with a score that depends on the likelihood that this model assigns to a held-out collection of unseen ground truth text data. The idea here is that, since we know that our held-out data was not seen by the model during training, and that it is well-formed, the higher the probability assigned by our model to this data, the higher the chances that it will, in general, generate well-formed samples. This principle is formalized through a metric called perplexity. Given a test set U_{test} consisting of samples

[10] We stress here that neither Greedy Sampling, nor Beam Search are guaranteed to produce the most likely sequence of tokens.

[11] Here, *well-formedeness* is relative to the data that we aim to learn. In that sense, a non-normalized text sample (*e.g. c'mon* instead of *come on*), is considered well-formed if it adheres to the distribution of the LM's training data, which could allow for non-normalized text.

$w = \{w_1, \ldots, w_{|w|}\}$, perplexity is calculated as follows:

$$PP(U_{test}) = p_\theta(U_{test})^{-\frac{1}{N}} \tag{2.11}$$

$$\text{s.t:} \quad p_\theta(U_{test}) = \prod_{w \in U_{test}} \prod_{i}^{|w|} p_\theta(w_i | w_{<i}) \tag{2.12}$$

$$N = \sum_{w \in U_{test}} |w| \tag{2.13}$$

In the above formula, N is the total number of words in our corpus, and it is used to normalize the inverse-probability score we measure. As shown in this formula, perplexity is a decreasing function of the token-wise probability $p_\theta(w_i|w_{<i})$. Therefore, lower perplexity means better language modeling performance. It is also important to note that perplexity *does not* account for the sampling strategy used at test time.

Now, in case we need to measure how close a model is to generating a specific output, a range of metrics have been developed in order to compare series of tokens in a linguistically meaningful way. The metrics we present here are metrics that were mainly aimed at measuring performance in the task of Machine Translation, and therefore designed to correlate with human judgment on sentence or phrase semantic similarity. To get the gist of how these metrics work, we describe, in what follows, the most commonly used metric: BiLingual Evaluation Understudy (BLEU; Papineni et al., 2001).

Given a generated (or candidate) sentence w and a set of reference translations $\{w^1_{ref}, \ldots, w^R_{ref}\}$ (*e.g.* alternative target translations for a single source sentence), this metric computes a *modified n-gram precision* of the n-grams present in the candidate, where n-grams are n-tuples of tokens. A standard n-gram precision measure consists in counting the number of n-grams in the candidate sentence that are present in a reference sentence, and then dividing the result by the total number of n-grams in this candidate. To see the problem with such a measure, observe the following example:

- **Candidate:** food food food food food.
- **Reference 1:** the dog ate its food.
- **Reference 2:** the canine consumed its nourishment.

Using standard precision, the above sentence that consists only in repeats of an n-gram that is present in a reference sentence (the unigram *food*) gets a perfect score. To deal with the above issue, BLEU modifies precision by clipping the total occurrence count for an n-gram in references by the maximum number of times it occurs in a reference. This

measure is formalized as follows:

$$p_n = \frac{\sum_{\mathcal{C} \in Candidates} \sum_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\sum_{\mathcal{C}' \in Candidates} \sum_{n\text{-}gram' \in C'} Count(n\text{-}gram')} \tag{2.14}$$

$$\text{s.t.:} \quad Count_{clip}(n\text{-}gram) = \min(Count, Max\_Ref\_Count) \tag{2.15}$$

In practice, BLEU is measured through a geometric mean of the above measures for different values of n, typically for $n \leq 4$.

In the same line of work, ROUGE [Lin, 2004] was engineered to maximally correlate with human judgment of summarization quality, and improvements over BLEU were brought about with METEOR [Banerjee and Lavie, 2005] and METEOR 1.5 [Denkowski and Lavie, 2014] which match words exactly, through stemming and through synonymy, and includes an explicit account for grammaticality through a chunking penalty.

More recently, efforts have been deployed to use neural representations as a continuous estimate of meaning, and to correlate similarity metrics between these representations (*e.g.* cosine similarities) with human judgments. A number of works have shown that state of the art correlations with human judgements could be obtained either with unsupervised representation learning techniques [Zhang et al., 2020, Kamal Eddine et al., 2022] or with supervised versions of these techniques where the representation is trained to be invariant to paraphrasing [Huang et al., 2021b].

2.5 Masked Language Modeling

As discussed earlier in this chapter, the term *Language Model* generally refers to models that assign probabilities to text samples. An exception to this rule is a recent class of models trained to perform a task dubbed *masked language modeling*. This task consists in recovering a sequence of tokens from a corrupted version of this sequence. More precisely, to train a Masked Language Model (MLM), one takes a sequence of tokens, replaces a proportion of the tokens with a special $[MASK]$ token, replaces another proportion of tokens with a random token, and trains a neural module p_θ to recover the original sequence using the following objective:

$$\underset{\theta}{\text{argmax}} \sum_{w \in L} \sum_{i}^{|w|} \log p_\theta(w_i | w_{corr}) \tag{2.16}$$

$$\text{s.t. :} \quad w_{corr} = corrupt(w) \tag{2.17}$$

where the *corrupt* operator applies the random replacements described above. Contrary to classical LMs, the above p_θ cannot be used to measure the probability of a sentence. In that sens, although it can be used to fill-in missing words in a sequence, it can not be used to generate sequences of tokens. In fact, it is important to notice that sampling

a token from $p_\theta(w_4|w_{corr,1} = t_1, w_{corr,2} = t_2, w_{corr,3} = t_3, w_{corr,4} = [MASK])$ does not yield the *next* token for the sequence $\{t_1, t_2, t_3\}$ but rather the *last* token since p_θ takes *bidirectional* information as a context. Therefore, using only previous tokens as a context for an MLM implicitly gives it the information that there are no other words in the sentence.

The concept of Masked Language Models (MLM) has been formulated and applied first in the work of Devlin et al. [2019] to train their Bidirectional Encoder Representations from Transformers (BERT). The objective was to obtain neural representations that feed off the entirety of the context in sentences through a self-supervised learning scheme, *i.e.* a learning scheme where the target information for inference is the input data itself. The intent behind this objective was to *transfer* the information channeled in these representations to other tasks. Transferring representations was popularized in NLP by Mikolov et al. [2013], and contextual representations were explored before MLMs (see for example Peters et al., 2018), but the work of Devlin et al. [2019] has shown that the ability of MLMs to factor context from the entire sentence enables contextual representations to display significantly higher transfer performance when applied to new tasks. The success of Devlin et al. [2019]'s model, BERT, was due to masked language modeling but also to the Transformer architecture. This architecture, which was highly popularized by BERT, is better explicited in Chapter 5, together with its implementation details, and some of the most prolific BERT variants that were also based on it.

2.6 Conclusion

Since the purpose of our work is to build interpretable neural LMs, we dedicated this chapter to introducing neural language modeling. First, we explained how Auto-regressive LMs can be formulated, and how they can be parameterized by learnable modules p_θ (§ 2.1). We then explained how, in addition to previous tokens, the probability assigned by p_θ to a token can be conditioned on other factors to explicitly model dependency on extrinsic attributes(§ 2.2).

Using LMs for generation requires sampling techniques, among which random sampling, Greedy Sampling and Beam Search are described in this Chapter(§ 2.3). Subsequently, we gave intuitive then formal descriptions of a few evaluation metrics, namely perplexity and BLEU, which are respectively used to measure the quality of unconstrained randomly generated samples, and the similarity of conditionally-generated samples with regard to a specific desired output(§ 2.4).

As a final point, we presented masked language modeling(§ 2.5), a training objective that deviates from classical language modeling in order to produce representations from a bidirectional textual context, and that provides the basis for modern high-performance NLP representation learning.

Chapter 3

Variational Autoencoders

Variational Autoencoders [Kingma and Welling, 2014] are models which leverage Deep Learning components by using the Autoencoder architecture [Rumelhart et al., 1985] to learn a generative model using Variational Inference [Jordan et al., 1999]. Contrary to what its name implies, its primary purpose is, therefore, not to learn encodings but rather to learn to generate observations that are likely to come from a given dataset $U = \{x_1, \ldots, x_{|U|}\}$. For a generative model $p(x) = \int_z p(x|z)p(z)dz$, where x is an observation and z is a latent variable with a known prior distribution p(z), Variational Inference can be used to approximate the posterior distribution $p(z|x)$, which is often intractable, by optimizing a different distribution $q(z|x)$. Using Neural Networks, or more precisely Autoencoders, this Variational Inference-based approach can be implemented using a neural encoder $q_\phi(z|x)$ and a neural decoder $p_\theta(x|z)$ to form what later came to be called Varational Autoencoders (VAEs). The presence of both an encoder and a decoder in VAEs led to a plethora of works that employ them outside of the strict generative modeling setup, namely for semi-supervised learning [Wolf-Sonkin et al., 2018, Habib et al., 2019, Corro and Titov, 2019, Chen et al., 2018a], or interpretable representation learning and controllable generation [Chen et al., 2019a, Cheng et al., 2020, Xu et al., 2020b, John et al., 2020, Bao et al., 2019, Huang and Chang, 2021b, Huang et al., 2021b].

This chapter is dedicated to presenting VAEs, the machine learning framework used throughout all of our contributions. We first describe the architecture it uses as a backbone, the Autoencoder architecture (§ 3.1). Then we explain how the encoder and decoder, together with a prior distribution on the latent variables, are used to formulate a lower-bound on the exact log-likelihood of a generative model (§ 3.2). This generative model requires a few implementation tricks to be learned efficiently which we explain in a dedicated section (§ 3.3). Language modeling with VAEs is discussed in Section 3.4, followed by the method used to evaluated perplexity for such models (§ 3.5). VAEs, when applied to language modeling, exhibit a particular type of failures called *posterior collapse* which we describe, together with common techniques to deal with it, in Section 3.6. Finally, we lay out the principles governing semi-supervised learning with VAEs (§ 3.7). The reasons why VAEs are capable of producing disentangled representations as well as the way they

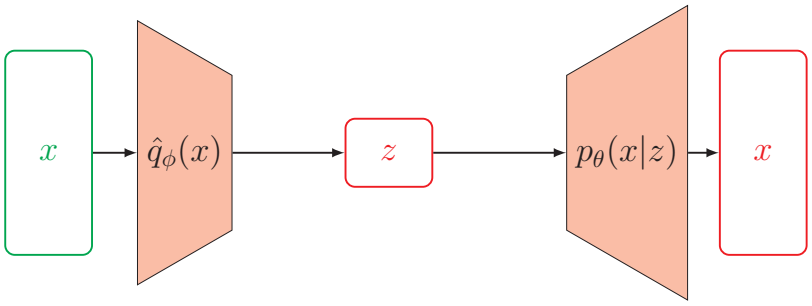


Figure 3.1: A diagram sketching the Autoencoder architecture. x is an observation, $\hat{q}_\phi(x)$ is an encoder and $p_\theta(x|z)$ is a decoder. Green denotes the ground truth variable, while red denotes inferred variables.

are used for that purpose are later detailed in Chapter 4.

3.1 The Autoencoder Architecture

Autoencoders [Rumelhart et al., 1985] are a class of Deep Learning models which aim to learn lossy compressions or *encodings* of high dimensional data through low dimensional representations. These models usually consist of an encoder function $\hat{q}_\phi$ which produces representations z from a samples x, and a decoder distribution p_θ which aims to reconstruct the input sample x from its representation z. This architecture is sketched in Figure 3.1.

Given observations x coming from a ground truth distribution p_{data}, the parameters ϕ and θ of Autoencoders can be learned by maximizing the following reconstruction objective:

$$\underset{\theta,\phi}{\operatorname{argmax}}\, \mathbb{E}_{x\sim p_{data}(x)} \left[\log p_\theta(x|z = \hat{q}_\phi(x))\right] \tag{3.1}$$

Notice that for *textual* observations, the above objective is exactly that of a conditional LM (*cf.* Eq. 2.8 in Section 2.2), where the conditional LM p_θ is jointly learned with q_ϕ, the module that provides the conditioning variables z. For the sake of the forthcoming explanations, we also point out the fact that the deterministic encoding function $\hat{q}_\phi(x)$ can be written as a conditional Dirac probability density function $q_\phi(z|x)$ such that:

$$q_\phi(z|x) = \begin{cases} +\infty & \text{if } z = \hat{q}_\phi(x) \\ 0 & \text{if } z \neq \hat{q}_\phi(x) \end{cases}$$

Which enables rewriting the reconstruction loss as:

$$\underset{\theta,\phi}{\operatorname{argmax}}\, \mathbb{E}_{x\sim p_{data}(x)} \mathbb{E}_{z\sim q_\phi(z|x)} [\log p_\theta(x|z)] \tag{3.2}$$

3.2 Theoretical Foundation of Variational Autoencoders

A generative model that generates observations x from a latent variable z with a known *prior* distribution, is modeled using this prior $p(z)$ and a *decoder* $p_\theta(x|z)$. Directly learning it through Likelihood Maximization is inefficient since the summation over all values of z to calculate $p_\theta(x) = \int_z p_\theta(x|z)p(z)dz$ is intractable in general[1]. For an observation x, Variational Inference uses a distribution $q(z)$[2] to formulate a lower-bound on the exact log-likelihood of the model $\log p_\theta(x)$. This lower-bound finds its origin in the following derivations:

$$\text{Given that } p_\theta(x) = \frac{p_\theta(x,z)}{p_\theta(z|x)} :$$
$$\log p_\theta(x) = \mathbb{E}_{z\sim q(z)} \log \frac{p_\theta(x,z)}{p_\theta(z|x)} \tag{3.3}$$

By splitting $p_\theta(x,z)$ to $p_\theta(x|z)p(z)$ and simply multiplying and dividing by $q(z)$, we obtain:

$$\mathbb{E}_{z\sim q(z)} \log \frac{p_\theta(x,z)}{p_\theta(z|x)} = \mathbb{E}_{z\sim q(z)} \left[\log \left(p_\theta(x|z)\frac{p(z)}{q(z)}\frac{q(z)}{p_\theta(z|x)}\right)\right] \tag{3.4}$$

Then, using the Kullback-Leibler Divergence operator defined by $\mathrm{KL}[q(z)||p(z)] = \mathbb{E}_{z\sim q(z)}\left[\log \frac{q(z)}{p(z)}\right]$, we obtain:

$$\mathbb{E}_{z\sim q(z)} \left[\log \left(p_\theta(x|z)\frac{p(z)}{q(z)}\frac{q(z)}{p_\theta(z|x)}\right)\right] = \mathbb{E}_{z\sim q(z)} \left[\log p_\theta(x|z)\right] - \mathrm{KL}[q(z)||p(z)]$$
$$+ \mathrm{KL}[q(z)||p_\theta(z|x)] \tag{3.5}$$

The above derivations lead to the equality:

$$\log p_\theta(x) - KL[q(z)||p_\theta(z|x)] = \mathbb{E}_{z\sim q(z)} \left[\log p_\theta(x|z)\right] - KL[q(z)||p(z)] \tag{3.6}$$

Since KL divergences are positive, the above equality allows writing the inequality that enables learning with VAEs[3]:

$$\log p_\theta(x) \geq \mathbb{E}_{z\sim q(z)} \left[\log p_\theta(x|z)\right] - \mathrm{KL}[q(z)||p(z)] = \mathrm{ELBo}(x;z) \tag{3.7}$$

The right-hand side of the above inequality, called the Evidence Lower-Bound (ELBo), is a tractable lower-bound to the exact log-likelihood of our model. According to Equation 3.6, maximizing this lower-bound either maximizes the log-likelihood of our model, or brings

[1] The integral can be made tractable if the support of $p(z)$ is small enough for summation, but such a choice of prior makes for an extremely weak generative model.

[2] We stress that each observation x is assigned with a dedicated $q(z)$. To lighten notations and to adhere to conventions adopted by other resource on this matter, q is not indexed with x.

[3] z, the latent variable, is marginalized throughout ELBo with expectations. Therefore, the value of ELBo does not depend on z. Nevertheless, we abuse notations and specify it as an argument in $\mathrm{ELBo}(x;z)$ to clarify the role of each variable in different instances of ELBo throughout the thesis.

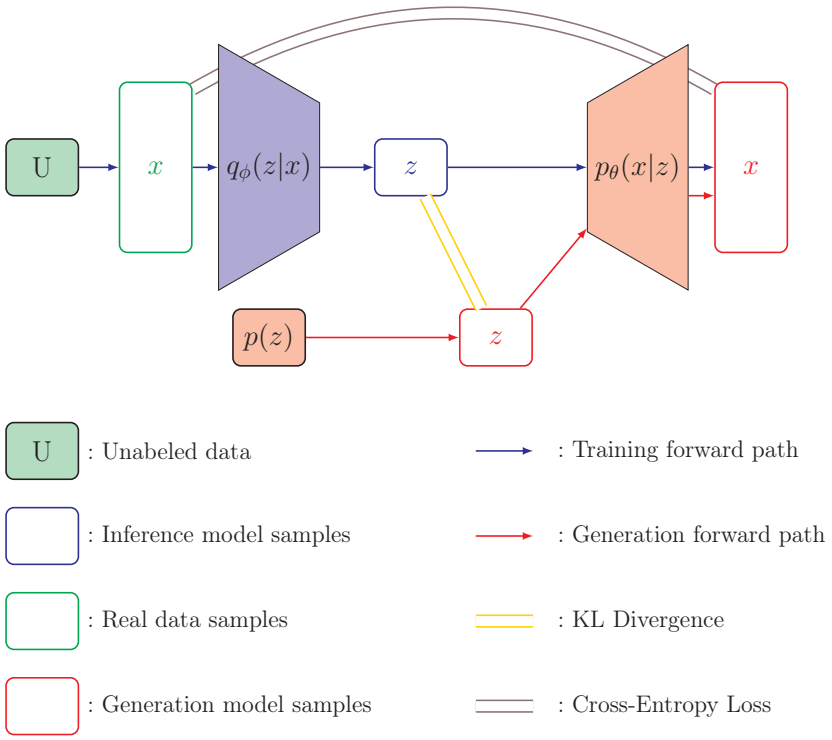


Figure 3.2: Variational Autoencoder Functioning scheme. Inference paths and components are colored in red, while components only used during training are colored in blue. Loss functions linking the different variables are sketched using links with two lines.

$q(z)$ closer to the true posterior $p_\theta(z|x)$. Since $q(z)$ is intended to mimic the true posterior, it is referred to as the *approximate posterior*.

Classical Variational Inference assumes the estimation of an approximate posterior $q(z)$ for each observation x. However, one can use a single model $q_\phi(z) = \int_x q_\phi(z|x)p_{data}(x)dx$ to learn approximate posteriors for all observations x. This is called Amortized Variational Inference (AVI; Kim et al., 2018).

Variational Autoencoders' learning is simply an application of the AVI learning scheme where $q_\phi(z|x)$, the encoder, and $p_\theta(x|z)$, the decoder, are neural networks which are learned through some form of Stochastic Gradient Descent (SGD).

After training, VAEs are used to generate samples with a two-step sampling process: First, a latent variable must be sampled from the prior distribution $p(z)$, then this latent variable is decoded using the decoder $p_\theta(x|z)$ to obtain a generated sample x.

Figure 3.2 summarizes the way a VAE is built, trained, and used for generation after training. It should be emphasized that, in the context of generative modeling, the input samples from the ground truth distribution together with the inference network (the blue components and the green components in Figure 3.2) are only used during training. At test time, only the prior distribution and the decoder (the red components in Figure 3.2) are used.

Figure 3.2 also illustrates the way ELBo is calculated. The first term in ELBo is a reconstruction term where the probability of the input sample $x \sim p_{data}(x)$ is evaluated according to $p_\theta(x|z)$, the output distribution on observations conditioned on a sample

from the encoder's distribution $q_\phi(z|x)$. The second term brings this encoder distribution, or approximate posterior, $q_\phi(z|x)$, closer to the prior distribution. In simple terms a VAE is trained to *reconstruct* input samples from latent codes that are *made likely to come from* the prior distribution.

3.3 Implementation Details

The objective ELBo requires calculating expectations over the distribution $q_\phi(z|x)$. This expectation is classically approximated using a Monte-Carlo estimate, *i.e.* an average of the term inside the expectation applied to a few samples from $q_\phi(z|x)$. Fortunately, empirical investigations [Kingma and Welling, 2014, Bowman et al., 2016] have shown that VAEs can be trained using single-sample estimations of the aforementioned expectations, and that learning with multiple samples displays no significant gain compared to learning with a single sample. This means that the reconstruction term in ELBo can be approximated during training as follows:

$$\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] \approx \sum_{i=1}^{N} \frac{1}{N} \log p_\theta(x|z_i) \approx \log p_\theta(x|z_0) \tag{3.8}$$

$$\text{s.t.} \quad z_0, z_1, \ldots, z_N \sim q_\phi(z|x) \tag{3.9}$$

Similarly, the Kullback-Leibler term can be efficiently estimated as follows:

$$\begin{aligned} &\mathrm{KL}[q_\phi(z|x)||p(z)] = \\ &\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{p(z)}{q_\phi(z|x)}\right] \approx \sum_{i=1}^{N} \frac{1}{N} \log \frac{p(z_0)}{q_\phi(z_0|x)} \approx \log \frac{p(z_0)}{q_\phi(z_0|x)} \end{aligned} \tag{3.10}$$

$$\text{s.t.} \quad z_0, z_1, \ldots, z_N \sim q_\phi(z|x) \tag{3.11}$$

Despite the fact that it is usable, the Monte-Carlo sampling approximation can be avoided for the Kullback-Leibler divergence term in ELBo in cases where a closed form is available for this divergence, *e.g.* if the prior $p(z)$ and the approximate posterior $q_\phi(z|x)$ are both Gaussian.

Although sampling enables efficiently estimating expectations, it requires the sampling operations to be differentiable in order to propagate the gradients to the encoder and to learn the parameters ϕ. This is made possible through this simple observation: given $\mu \in \mathbf{R}^d$ and $\sigma \in \mathbf{R}^{d*d}$ if z, a d-dimensional random variable, follows a standard normal distribution $\mathcal{N}(0, I_d)$, then $\sigma z + \mu$ follows $\mathcal{N}(\mu, \sigma)$. Accordingly, the expectation of a function $f(z)$ over a Gaussian distribution $q_\phi(z|x)$ with mean vector $\mu(x)$ and standard

deviation $\sigma(x)$ can be reformulated as follows:

$$\mathbb{E}_{z\sim q_\phi(z|x)}[f(z)] = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(\sigma(x) * z + \mu(x))] \tag{3.12}$$

Using the above formula decouples the parameterization from the sampling procedure and introduces it through differentiable operations (addition and multiplication). This trick, called the *reparameterization trick* [Kingma and Welling, 2014], is applicable to any distribution from the *location-scale* family (Gaussian, Laplacian, ...). Posterior to the initial work of Kingma and Welling [2014] on Variational Autoencoders where the reparametrization trick was introduced, differentiable sampling schemes have also been developed for Categorical distributions such as the Gumbel-Softmax Trick [Jang et al., 2017], and direct optimization through argmax [Lorberbom et al., 2019].

Besides the above implementation tricks, a few techniques have also been developed to reduce the variance of the stochastic gradient estimation during the optimization process. Namely, throughout the experiments carried in this thesis, we use a technique called Sticking The Landing (STL-ELBo; Roeder et al., 2017) which helps reduce the variance of the gradients estimated during the learning process. The techniques stems from a derivation on the gradient of ELBo which identifies a term which is null in expectation. To identify this term, let us first derive ELBo as follows:

$$\begin{aligned}
\text{ELBo} &= \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - KL[q_\phi(z|x)||p(z)] && (3.13)\\
&= \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z) + \log p_\theta(z) - \log q_\phi(z|x)\right] && (3.14)\\
&= \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(z|x) + \log p_\theta(x) - \log q_\phi(z|x)\right] && (3.15)
\end{aligned}$$

where the last line exploits the equality $p_\theta(x|z)p(z) = p_\theta(x,z) = p_\theta(z|x)p_\theta(x)$. In the following, we calculate the total derivative of the expression inside the expectation operator w.r.t ϕ, which yields partial derivatives w.r.t ϕ and z since z is a function of ϕ:

$$\begin{aligned}
&\frac{d}{d\phi}\left[\log p_\theta(z|x) + \log p_\theta(x) - \log q_\phi(z|x)\right] && (3.16)\\
=&\frac{\partial}{\partial z}\left[\log p_\theta(z|x) - \log q_\phi(z|x)\right]\frac{\partial z}{\partial \phi} - \frac{\partial}{\partial \phi}\log q_\phi(z|x) && (3.17)
\end{aligned}$$

In the last line, the expectation of the last term can be proven null as follows:

$$\begin{aligned}
&\mathbb{E}_{z\sim q_\phi(z|x)}\frac{\partial}{\partial\phi}\log q_\phi(z|x) = \int_z q_\phi(z|x)\frac{\partial}{\partial\phi}\log q_\phi(z|x)dz && (3.18)\\
=&\int_z q_\phi(z|x)\frac{1}{q_\phi(z|x)}\frac{\partial}{\partial\phi}q_\phi(z|x)dz = \int_z \frac{\partial}{\partial\phi}q_\phi(z|x)dz && (3.19)\\
=&\frac{\partial}{\partial\phi}\int_z q_\phi(z|x)dz = \frac{\partial}{\partial\phi}1 = 0 && (3.20)
\end{aligned}$$

Since this term has a null expectation, it can be dropped from the gradient estimator

without biasing it. The STL-ELBo technique consists in dropping this term, which leads to better log-likelihood results than the standard ELBo estimator, as shown by Roeder et al. [2017].

3.4 Language Modeling with a VAE

VAEs were first explored as LMs by Bowman et al. [2016]. Prior to this work, Neural LMs were mainly tackled through purely auto-regressive generation schemes such as the one presented in Section 2.1. Purely auto-regressive generation schemes, mainly implemented in recent works with LSTMs [Melis et al., 2020] or Transformers [Radford et al., 2019], achieve low perplexity values and model long-term dependencies through step-wise updates to a context vector, which does not clearly separate sequence-level factors of variation for word-level factors of variation. When need arises, the sequence-level factors are usually modeled through the introduction of conditioning variables to the generative model, where the model is trained with supervised learning as was explained in Section 2.2. However, ELBo provides a means to train a generative LM in the form $p_\theta(x) = \int p_\theta(x|z)p(z)dz$ while waving the need for any specifics on z beyond the prior $p(z)$. Therefore, in order to train generative models with explicit sequence-level representations z, Bowman et al. [2016] explored the VAE option.

Bowman et al. [2015] trained VAEs for language modeling, using single-layer LSTMs as encoders and RNNs conditioned on the latent codes as decoders. They obtained competitive results with RNN LMs on the Penn Treenbank [Marcinkiewicz, 1994] dataset. Their work also led to a few interesting observations on such VAE LMs. They first observed that, given that a VAE encoder is trained to concentrate all representations in the same vicinity through the KL-Term, and to generate well-formed sentences from this vicinity through the reconstruction term, VAEs had what they called a *smooth latent space*. This smoothness manifests in the fact that intervening on a sample z (*e.g.* resampling components or interpolating with another vector) from an encoded sentence and decoding the resulting latent vector resulted in well-formed sentences. In contrast to this, operating on representation from classical Sequence-to-Sequence Autoencoders [Sutskever et al., 2014] in their latent spaces often leads to malformed decoded sentences. To illustrate this smoothness, Bowman et al. [2016] produce sentences by decoding latent codes corresponding to progressive linear interpolations of the representations of two sentences[4]. The original sentences and the sentences generated from the progressive interpolation are displayed in Table 3.1.

The sentences displayed in Table 3.1 clearly show progressive change from the first sentence to the second sentence. This smoothness property is especially valuable when it comes to combining characteristics from different sentences in the context of controllable generation.

A second valuable observation highlighted by this work was a phenomenon dubbed

[4]This process is often refered to as a *homotopy*.

i went to the store to buy some groceries .
i store to buy some groceries .
i were to buy any groceries .
horses are to buy any groceries .
horses are to buy any animal .
horses the favorite any animal .
horses the favorite favorite animal .
horses are my favorite animal .

Table 3.1: Example sentence interpolations using VAE latent variables from Bowman et al. [2016]. Original sentences constituting the edges of the linear interpolation are given in bold.

posterior collapse, where all the posterior distributions $q_\phi(z|x)$ converge (or *collapse*) to the prior distribution $p(z)$, and therefore become uninformative on the input x. This phenomenon stems from the fact that auto-regressive decoders used in VAE LMs are powerful enough to learn good LMs without the need for latent variables. Solutions to avoid this problem are detailed in Section 3.6

3.5 A Tighter Lower-Bound to the Log-Likelihood with Importance Weighting

VAEs allow optimizing a generative model via ELBo, a lower-bound to the exact log-likelihood. ELBo is a convenient strategy when it comes to training, but testing generative models often requires estimating the exact likelihood of data according to a model. In fact, as explained in Section 2.4, assessing the generative capabilities of an LM requires estimating its perplexity, which is based on the exact likelihood measure. In order to provide such a measure for VAEs, so as to be able to compare their generative capabilities to standard Maximum Likelihood-based LMs, one needs a tight estimate of the exact likelihood.

Posterior to Kingma and Welling [2014]'s work on VAEs, the following Importance Weighting-based lower-bound to the exact log-likelihood of a VAE has been developed by Burda et al. [2016]:

$$\log p_\theta(x) \quad \geq \quad \mathbb{E}_{z_1,\dots,z_k \sim q_\phi(z|x)} \left[\log \frac{1}{k} \sum_{i=1}^{k} \frac{p_\theta(x, z_i)}{q_\phi(z_i|x)} \right] \quad = \quad \text{IWO}(x; z; k) \quad (3.21)$$

VAEs using this Importance Weighted Objective (IWO) during training are referred to as Importance Weighted Autoencoders (IWAEs). Notice that this objective is equal to ELBo when it uses a single sample (*i.e.* $k = 1$). More importantly, Burda et al. [2016]

provided a proof establishing that this objective satisfies the following inequality:

$$\log p_\theta(x) \geq \text{IWO}(x; z; k+1) \geq \text{IWO}(x; z; k) \tag{3.22}$$

and that, under mild assumptions, $\text{IWO}(x; z; k)$ converges to $\log p_\theta(x)$ when k goes to infinity. This means that using more importance samples (*i.e.* a higher k) allows monotonously approaching an exact estimate of the log-likelihood, and therefore approaching the exact value of perplexity[5] for a language model.

Other tight lower-bounds to the exact log-likelihood were developed after IWAE, such as the Thermodynamic Variational Objective (TVO; Masrani et al., 2019), but Importance Weighting remains the standard method to assess VAE-based LM perplexities.

3.6 Dealing with Posterior Collapse

As mentioned in Section 3.4, VAEs suffer from posterior collapse, a phenomenon where the VAE LM converges to a local minimum where the LM is only learned through the autoregressive decoder, and the distributions $q_\phi(z|x)$ parameterized by the encoder all converge exactly to $p(z)$ minimizing the Kullback-Leibler divergence term in ELBo. A first solution to this problem was proposed in the work of Bowman et al. [2016], and consists in *jump-starting* the encoder by multiplying the KL term by $\beta = 0$ for a few optimization steps, so that the whole network is only optimized for reconstruction. The KL term is then slowly introduced over a number of *annealing* steps by linearly increasing β to 1. Another trick they implemented which mitigated posterior collapse was *word-dropout*: A technique where a proportion (set to 40% in their work) of the previous words in the auto-regressive generation scheme of decoders is replaced with a null vector during training to enforce reliance on the latent codes.

In the later work of Kingma et al. [2016], this solution was improved upon through a strategy called Free-bits strategy or KL-thresholding strategy. Given that, for a d-dimensional latent variable $z = \{z_1, \ldots, z_d\}$, the prior $p(z)$ and the approximate posterior $q_\phi(z|x)$ are most often respectively set to a standard normal distribution $\mathcal{N}(0, I_d)$ and to a diagonal Gaussian distribution $\mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$, The KL-term can be written as the sum of dimension-wise KL divergences between prior and approximate posterior distributions, *i.e.* $\text{KL}[q_\phi(z|x), p(z)] = \sum_i^d \text{KL}[q_\phi(z_i|x), p(z_i)]$. The KL-thresholding strategy consists in minimizing these dimension-wise KL terms as part of the ELBo optimization only down to a global threshold γ.

Other works have proposed diverse solutions to the posterior collapse problem throughout the years such as a cyclic annealing scheme for the KL term [Fu et al., 2019], or aggressive optimization phases that focus on the encoder [He et al., 2019], but Li et al. [2019] have shown that simply combining KL annealing with KL-thresholding leads to the most effective fix to the posterior collapse problem while still requiring minimal mod-

[5]Since perplexity is a decreasing function of likelihood, our estimate here would be an upper-bound to the perplexity.

ification to the original VAE learning scheme. In fact, the modification simply consists in an ELBo that takes the following form:

$$\text{ELBo}(x; z) = \mathbb{E}_{z\sim q(z)}\left[\log p_\theta(x|z)\right] - \beta \sum_{i}^{d} \max(\gamma, \text{KL}[q_\phi(z_i|x)||p(z_i)]) \tag{3.23}$$

where γ is fixed, and β is used for the progressive introduction of the KL term to the optimization procedure as described above. It is worth noting that a common practice in VAE-based language modeling to counteract posterior collapse is simply to lower the final value of β so as to allow $q_\phi(z|x)$ to drift further from $p(z)$ in order to better optimize the model for reconstruction and therefore to absorb more information about x. This is especially common in works that employ VAEs as *regularized* Autoencoders for controllable generation (*e.g.* Chen et al., 2019b).

More recent works like those of Wu et al. [2020] and Menon et al. [2022] propose solutions that further mitigate posterior collapse, but they both employ auxiliary networks with sizes similar to that of the original generative model. Since designing competitive LMs in the Deep Learning era is a memory sensitive endeavor, we stick to the above combination proposed by Li et al. [2019] for the works presented in this thesis as it constitutes a suitable middle ground between effectiveness and memory efficiency.

3.7 Semi-Supervised Learning with VAEs

Standard supervised learning considers the setup where an inference module $q_\phi(y|x)$ is tasked with learning to predict labels y from observations x using a dataset of labeled samples $L = \{(x_1, y_1), ..., (x_{|L|}, y_{|L|})\}$. The semi-supervised learning setup however, also makes use of an auxiliary set of unlabeled data points $U = \{x'_1, ..., x'_{|U|}\}$ which are typically available in much larger quantities than their labeled counterparts. The use of VAEs in semi-supervised learning has first been explored by Kingma et al. [2014].

The idea here, is to use the VAE encoder as a classifier by introducing the target label as a latent variable to the VAE-based generative model. Specifically, besides the usual unobserved latent variable z, the semi-supervised VAE framework also uses a partially-observed latent variable y. These variables are typically modeled as independent latent variables such that $q_\phi(y, z|x) = q_\phi(y|x)q_\phi(z|x)$ and $p_\theta(x) = \int p_\theta(x|y, z)p(y)p(z)$. The encoder $q_\phi(y|x)$ serves both as the inference module for the supervised task, and as an approximate posterior (and encoder) for the y variable in the VAE framework. Accordingly, y is learned on data from U in a similar manner to z, *i.e.* as an unobserved latent variable as part of the VAE generative model. But when presented with data from L, the learning procedure uses y both as a target for supervised learning with the encoder $q_\phi(y|x)$ and as an observed conditioning variable for the decoder $p_\theta(x|y, z)$ on L. Formally, the training objective $\mathcal{J}^\alpha$ [Kingma et al., 2014] for a Semi-Supervised VAE (SSVAE) is

expressed as follows:

$$\mathcal{J}^{\alpha} = \sum_{(x,y)\in L} \Big(\text{ELBo}((x,y);z) + \alpha \ \log q_{\phi}(y|x) \Big) + \sum_{x \in U} \text{ELBo}(x;(y,z)) \tag{3.24}$$

where the first argument of ELBo is the set of observed variables, and the second argument is the set of unobserved variables. As can be seen in the formula above, for samples coming from L, $q_{\phi}(y|x)$ is trained to predict y using a Cross-Entropy[6] objective that is weighted with regard to the remaining terms of the overall objective using a scalar α.

3.8 Conclusion

This chapter is an introduction to the main technical framework for this thesis: Variational Autoencoders. We explained the motivation behind generative modeling with VAEs in general, and Language Modeling with VAEs in particular. Starting from the standard Autoencoder architecture (§ 3.1), we explained the additions and modifications required to obtain the basic components of a VAE, and to derive its objective ELBo (§ 3.2). To give better practical insight into VAEs, we explained in Section 3.3 the basic tricks required to implement it and to train it. Given the NLP-specific context of this thesis, we also summarized the main findings of the first work on VAE-based LMs (§ 3.4). We also layed-out the IWO, a tighter lower-bound to the exact log-likelihood which constitutes the basis for estimating perplexity in order to measure language modeling performance (§ 3.5). The particular case of language modeling with VAEs suffers from an issue called posterior collapse, which we explained and for which we described a few solutions while pointing to the solution we retain for the works described in this thesis (§ 3.6).

Finally, we broke down semi-supervised learning with VAEs in the final section of this chapter (§ 3.7) so as to provide the prerequisites for our semi-supervised learning-related contribution. The second part of our contributions, which is dedicated to disentanglement, relies on the disentanglement capabilities of VAEs. As this is better explained following a few introductory notions, we elaborate on the relation between VAEs and disentanglement in the chapter dedicated to disentanglement (Chapter 4).

[6]Recall, here, that the Cross-Entropy objective is an application of Maximum-Likelihood Estimation. In fact, maximizing the likelihood of ground truth labels according to a classifier is equivalent to minimizing the Cross-Entropy between the ground truth label distribution, and the label distribution defined by the classifier.

Chapter 4

Disentangled Representation Learning

A neural network is a function f_θ with parameters θ, where the parameters are estimated so as to *learn* a relation between an input to the function and a desired output. This function is most often built as a composition of functions such that $f_\theta = f^1_{\theta_1} \circ f^2_{\theta_2}, \ldots, \circ f^n_{\theta_n}$ where each $f^i_{\theta_i}$ is called a *layer* and n is the number of layers in the neural network f_θ. The individual $f^i_{\theta_i}$ have multidimensional outputs, where each output is called a neuron, and any combination of neurons (*e.g.* a layer, part of a layer, a concatenation of layers) is called a *neural representation*. Neural representations have shown great promise as automatic feature extractors in and outside of NLP [Mikolov et al., 2013, Simonyan and Zisserman, 2014, He et al., 2016b, Devlin et al., 2019]. However, aside from output neurons, it is difficult to separate and identify understandable concepts in neural representations. Investigating methods which yield neural representations with identified interpretable concepts is the very purpose of disentanglement. More precisely, disentangled representation learning aims at designing methods which produce neural representations where each dimension, or set of dimensions, relates significantly to an understandable concept.

Interest in disentangled neural representation learning started emerging in the early 2010's where, for instance, Bengio et al. [2013] argue in their review of representation learning that it is crucial for the future of AI to disentangle understandable factors of variation within neural representations. It persists as a crucial research direction in recent works such as Rudin et al. [2022] which lists supervised and unsupervised disentanglement within the ten great challenges for interpretable machine learning. Besides interpretability, disentanglement is also sought as a means to improve sample complexity under the intuition that reducing inputs to a minimalistic set of human-validated interpretable concepts should improve generalization. This intuition was empirically confirmed by a number of studies such as the large-scale experiments carried by Locatello et al. [2020b] which show that disentangled representations improve sample efficiency and generalization under covariate shift(*i.e.* a change in the distribution of observations).

From a practical point of view, disentanglement is induced either with supervised learning where neural representations are constrained to be informative on a given factor, or

through unsupervised learning which is the paradigm chosen for this thesis. An underlying assumption to unsupervised disentanglement, is that the data can be factored into a set of *independent* generative factors. This was the main guiding principle for the design of unsupervised disentanglement methods, and also the main reason behind the successes of VAEs at disentanglement.

As a starting point, we describe the mechanisms inside VAEs that encourage obtaining independent generative factors (§ 4.1). In the following section, we clarify the relation between the factorization induced by VAEs and disentanglement using a few recent theoretical results(§ 4.2). Subsequently, we describe the few common ways inductive bias is introduced to neural networks in order for them to exhibit disentanglement, specifically in a supervised or in an unsupervised fashion(§ 4.3). Moving on to evaluation, in Section 4.4, we summarize the intuition behind measures of disentanglement, and detail a few measures that serve here as an example, and later in the thesis (Chapter 8) as the basis for disentanglement measures of our own design. Finally, we give in Section 4.5 an overview of the *disentanglement in NLP* landscape with an emphasis on a few shortcomings of the current literature in regard to which we aim to contribute in this thesis.

4.1 Finding Independent Generative Factors with VAEs

VAEs are a natural candidate method for disentanglement as their primary purpose is to relate observations to latent variables. The conditional or marginal distributions of these latent variables are set by the practitioner, which most often chooses them to be independent. In fact, the most common VAE is built with a standard Gaussian prior $p(z)$ and a diagonal Gaussian approximate posterior $q_\phi(z)$. To the best of our knowledge, choosing Gaussians for both distributions mostly stems from the fact that practitioners usually default to Gaussians when no information is available about the underlying distribution of a variable. This default choice also enables using a closed form of the KL divergence instead of its Monte-Carlo estimate. It can be argued, for the standard Normal distribution often set to be the prior, that it works as a regularization on latent variables [Chen et al., 2018a, Wolf-Sonkin et al., 2018, Yacoby et al., 2020], similar to a Ridge regularization [Hoerl and Kennard, 1970]. As for the posterior distribution, the choice of a diagonal Gaussian was due to computational gains, since it only requires estimating the diagonal elements of the covariance matrix instead of the entire matrix. Given these design choices, latent variable modeling in VAEs aims to retrieve a set of independent generative factors from the observations, which makes it an intuitive choice for disentanglement.

The first approach to disentanglement using VAEs is called β-VAE [Higgins et al., 2017], as it simply consists in adding a scalar β to scale the KL term in ELBo (Eq. 3.7):

$$\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \beta\,\mathrm{KL}[q_\phi(z|x)||p(z)] \tag{4.1}$$

The argument behind this modification is fairly straightforward: The KL term pushes the

distribution of the encoded representations $q_\phi(z|x)$ towards $p(z)$, which is most often a standard Normal distribution that encourages the different dimensions in $q_\phi(z|x)$ to be independent. Therefore, β-VAE allows choosing a compromise between high disentanglement with a high β, or high reconstruction fidelity with a low β.

The subsequent work of Chen et al. [2018c] further pinpoints the source of disentanglement in ELBo through a decomposition of its KL term. Assuming that observations x in the training set are indexed with an integer $n \in \{1, \dots, N\}$, we can define a uniform distribution $p(n)$ over that range, and subsequently consider the distributions $q_\phi(z|n) = q_\phi(z|x_n)$, $q_\phi(z, n) = q_\phi(z|n)p(n) = q_\phi(z|n)\frac{1}{N}$ and $q_\phi(z) = \sum_1^N q_\phi(z|n)p(n)$. These distributions can be used to write the following decomposition[1]:

$$\begin{aligned} \mathbb{E}_{n\sim p(n)} KL[q_\phi(z|n)||p(z)] =& KL[q_\phi(z,n)||q_\phi(z)p(n)] && ① \\ &+ KL[q_\phi(z)||\prod_j q_\phi(z_j)] && ② \\ &+ \sum_j KL[q_\phi(z_j)||p(z_j)] && ③ \end{aligned} \tag{4.2}$$

The decomposition of the KL term of ELBo in equation 4.2 displays three components: ① The index-code Mutual Information, which is the mutual information between inputs x (indexed by n in the equation) and latent variables z; ② The Total-Correlation (TC), which is a measure of dependence between components of the latent variable vector; ③ The dimension-wise KL divergence. Chen et al. [2018c] argue that component ②, TC, is responsible for disentanglement, since it measures the dependence between components of latent vectors, and propose to control a weight on this component instead of the entire KL term, leading to the TC-VAE objective.

Driven by the same intuition, other objectives were designed in order to force independence between the components of latent vectors, namely Annealed VAE [Burgess et al., 2017], FactorVAE [Kim and Mnih, 2018], and DIP-VAE [Kumar et al., 2018]. However, the large-scale experiments carried by Locatello et al. [2020a] have shown that no objective from the above encourages disentanglement significantly more than the others, and that, in general, random seeds influence disentanglement results more than the choice of method. Therefore, for its simplicity, β-VAE is the most commonly chosen objective in recent disentanglement works. It is also the objective chosen to carry out the works described in this thesis.

4.2 About the Alignment Between Independent Generative Factors and Understandable Concepts

Crucially, the above methods *only* enable relating observations x to independent generative factors z. However, does this mean that such methods will converge to *THE* target independent generative factors ? As will be argued below using Figure 4.1, the answer is

[1] The full derivation leading to this decomposition can be found in Appendix A.1.

"no".

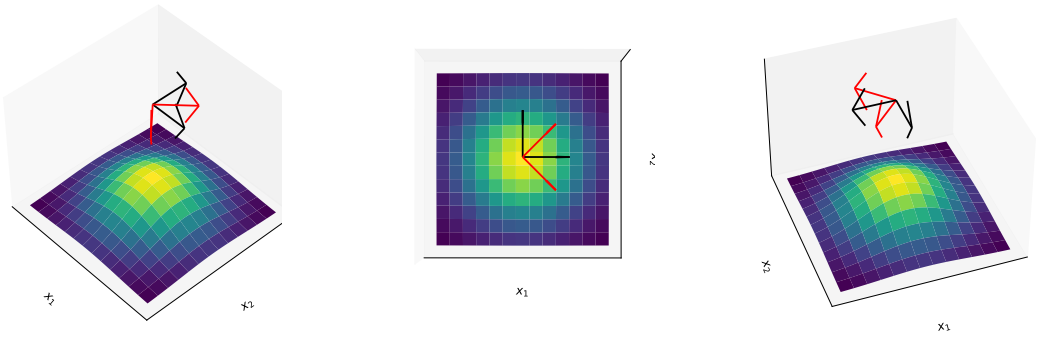


Figure 4.1: Different perspectives on a 2D Gaussian distribution. Arrows in black represent directions x_1 and x_2, which are two independent axes of variation for this Gaussian distribution. Arrows in red represent directions $(x_1 + x_2)$ and $(x_1 - x_2)$, which also represent independent axes of variation.

Figure 4.1 is a plot of the density of a 2D standard Normal distribution. Coordinates x_1 and x_2 of 2D samples from this distribution have independent distributions. Therefore, x_1 and x_2 represent independent generative factors for samples from this distribution. However, notice that $(x_1 + x_2)$ and $(x_1 - x_2)$ also have independent realizations, and thus equally represent independent generative factors for samples from this distribution. The above simple case demonstrates that simply seeking independent generative factors *cannot alone* yield generative factors that align with a predefined set of such factors. This rationale is the basis of the *unsupervised disentanglement impossibility* result presented in the work of Locatello et al. [2020a].

Through a formal proof that unsupervised factorization into independent generative factors cannot align different latent variables with identifiable disentangled generative factors, Locatello et al. [2020a] argue that disentanglement requires an additional form of inductive bias in order to align latent variables with identifiable generative factors.

Posterior to that work, Rolinek et al. [2019] presented yet another important result: standard VAE implementation guidelines induce a behavior that mimics Principled Component Analysis (PCA): As mentioned earlier, the most commonly used distributions for a VAE are a standard Normal prior, and a diagonal Normal approximate posterior. Rolinek et al. [2019] show that a PCA-like behavior is displayed by VAEs and that it is caused by diagonal posteriors. Most importantly, PCA factors observations into independent axes of variation, and greedily select the highest direction of variation at each factorization step. Contrary to a plain search of independent factors, this process converges to a fixed set of generative factors (up to a permutation operation).

Rolinek's result requires the covariance matrix of the different generative factors to have distinct singular values. This means that the different generative factors must contribute differently to the overall variance of the samples. In fact, for generative factors with equal contribution to the overall variance, *e.g.* x_1 and x_2 for the isotropic Gaussian in Figure 4.1, Locatello's impossibility result stands unquestioned. In practice, Rolinek shows

that disentanglement works out better when the gap in variance is higher between generative factors, which partly explains the difference in disentanglement performance across different datasets. In light of these observations, and as concluded by Locatello, VAEs encourage disentanglement but require additional inductive bias for disentanglement either from the model or from the data. The nature of such inductive bias is explicited in the next section.

4.3 Disentanglement as a Result of Inductive Bias

In this section, we describe different methods to implement bias inducing separation in the latent representations of neural networks. The most explicit method is of course a form of supervision. To implement separation, a latent variable z is usually split into two latent variables z_1 and z_2, where we try to associate one of the variables (*e.g.* z_1 in what follows) to the target factor of variation y. Given generative model $p_\theta(x) = \int_{z_1,z_2} p_\theta(x|z_1,z_2)p(z_1,z_2)dz_1dz_2$ with approximate posterior $q_\phi(z_1,z_2|x)$ and labeled data L aligning observation x with generative factors y, the latent variables z_1 can be made informative about generative factor y simply by means of the objective:

$$\sum_{(x,y)\in L} \text{ELBo}(x;(z_1,z_2)) + \alpha\mathbb{E}_{z_1\sim q_\phi(z_1|x)}[q_\phi(y|z_1)] \tag{4.3}$$

where $q_\phi(y|z_1)$[2] is an additional inference module relating z_1 to y, and α is a weighting coefficient. The objective described in Equation 4.3 encourages z_1 to be fully informative on y. Given that VAEs also encourage latent variables to be independent, encouraging all the information about y to be in z_1 also discourages z_2 from including any information on y.

Alternatively, bias towards disentanglement can be induced through the way a model is built. To better understand this, we take the example of HoloGAN [Nguyen-Phuoc et al., 2019], a model for natural image generation. During generation, this model generates a 3D structure from a latent vector z, chooses an angle θ, then projects the 3D structure on a plane according to θ in order to obtain a 2D image. This model is trained as a Generative Adversarial Network (GAN; Goodfellow et al., 2014) to generate natural images. Without any information on the latent 3D structure or its orientation in samples from the training set, this model naturally learns to decouple elements represented in natural images from their pose. To better summarize the idea, Figure 4.2 shows a graphic by Nguyen-Phuoc et al. [2019] as well as a few samples from their generative model.

HoloGAN's example shows that disentanglement can be induced in neural models given a *shared inductive bias* between the model and the data it is trained on. Namely here, the model infers 3D structures and projects them on 2D planes in accordance with the data which consists in 3D structures projected on 2D planes.

[2]If the random variable z_1 has the same support as y, z_1 can be directly trained to take the values of y by setting $q_\phi(y|z_1)$ to an identity function. In this case, one can even use the Semi-Supervised VAE objective for labeled data from Equation 3.24 in Section 3.7.

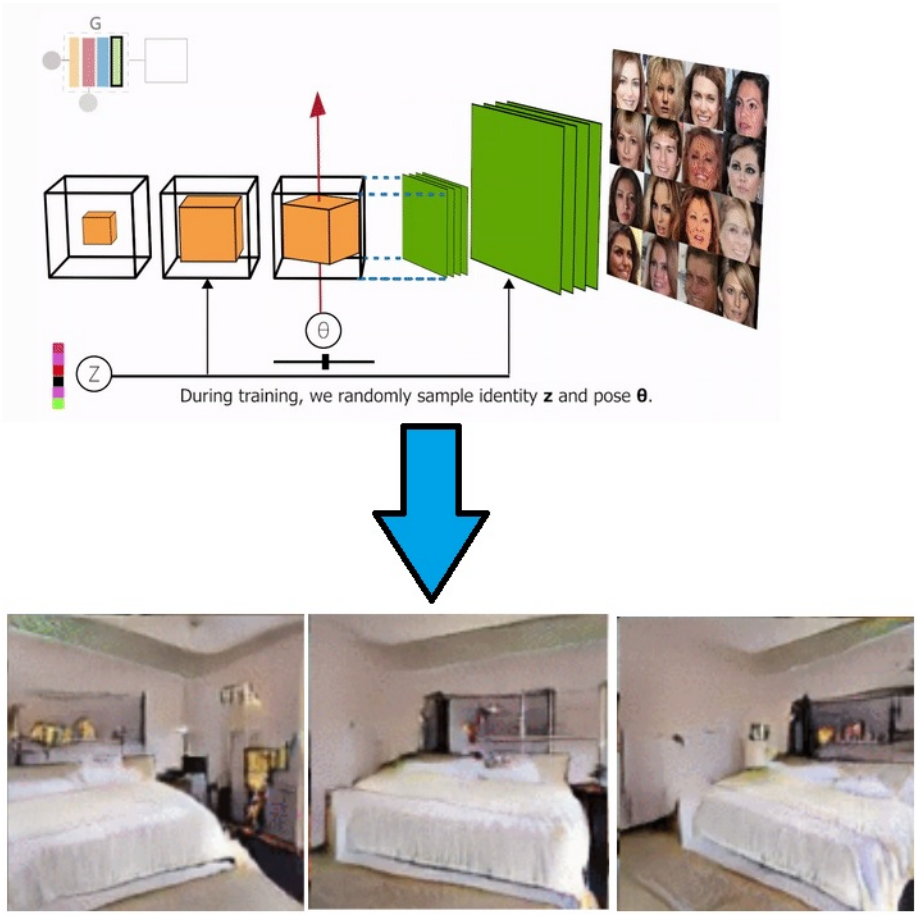


Figure 4.2: A graphic by Nguyen-Phuoc et al. [2019] summarizing the generation steps in HoloGAN together with a few generated sample at different angles.

4.4 Measuring Disentanglement

Perfectly disentangled neural representations are supposed to display 3 characteristics [Ridgeway and Mozer, 2018, Eastwood and Williams, 2018]:

1. *Modularity/Disentanglement:* Each portion[3] of the latent code must capture at most one factor of variation.

2. *Compactness/Completeness*: Each factor of variation must be captured by at most one portion of the latent code.

3. *Informativeness/Explicitness*: The representations must be maximally informative on the factors of variation.

[3] *Portions* here emphasizes the fact that one can aim for disentanglement along distinct single dimensions in the neural representations, or a multi-dimensional partition of neural representations.

A generic measure for disentanglement: Multiple *generic* metrics have been developed in the literature with the above characteristics in mind [Higgins et al., 2017, Kim and Mnih, 2018, Ridgeway and Mozer, 2018]. As an example, we detail the design of Mutual Information Gap (MIG; Chen et al., 2018b) which is fairly generic, accounts for axis alignment between latent variable dimensions and generative factors, and requires no *post-hoc* classifier training. To define this metric, Chen et al. [2018b] first define the joint distribution $q_\phi(z_j, v_k) = \sum_{n=1}^{N} p(v_k)p(n|v_k)q_\phi(z_j|n)$, where $v_k \in \{v_1, \dots, v_K\}$ is a generative factor, n is the index of a ground truth sample among N available samples, and $z_j \in \{z_1, \dots, z_J\}$ is a component of the latent vector. Given that joint distribution, the following empirical estimate of mutual information between latent dimension z_j and generative factor v_k is defined :

$$I_n(z_j; v_k) = \mathbb{E}_{z_j, v_k \sim q(z_j, v_k)} \left[log \sum_{n \in \mathcal{X}_{v_k}} q_\phi(z_j|n)p(n|v_k) \right] + H(z_j) \tag{4.4}$$

where $\mathcal{X}_{v_k}$ is the support of $p(n|v_k)$; *i.e.* the available samples verifying the sampled v_k. Using this estimator, MIG sums over all generative factors the gap between the highest calculated I_n with regard to a component of the latent vector and the second highest of such measures, normalized by the entropy of the present generative factor:

$$\frac{1}{K} \sum_{k=1}^{K} \frac{1}{H(v_k)} \left(I_n(z_{j^{(k)}}; v_k) - \max_{j \neq j^{(k)}} I_n(z_j; v_k) \right) \tag{4.5}$$

where $j^{(k)} = \text{argmax}_j \, I_n(z_j; v_k)$. For each generative factor v_k, the corresponding component in the above sum is high if it has high mutual information with *a single latent dimension*, since the value of this component is penalized by the second highest mutual information. Ideally, the component is equal to 1 if the most informative latent variable is perfectly informative on v_k (meaning $I_n(z_{j^{(k)}}; v_k) = H(v_k)$) and if all other latent variables contain no information on v_k (meaning $\max_{j \neq j^{(k)}} I_n(z_j; v_k) = 0$).

Notice that, among the 3 criteria explicited above, MIG only captures Compactness and Informativeness. In fact, a representation where only one latent dimension z_j captures all the information from all generative factors v_k would obtain a perfect score of MIG=1. Therefore, to complement MIG on Modularity, Li et al. [2020c] define MIG-sup, which simply differs with MIG through a partial[4] permutation of the roles of latent variables and generative factors:

$$\frac{1}{J} \sum_{j=1}^{J} \frac{1}{H(v_k)} \left(I_n(z_j; v_{k^{(j)}}) - \max_{k \neq k^{(j)}} I_n(z_j; v_k) \right) \tag{4.6}$$

[4]The permutation is only partial in that the normalization in MIG-sup is still done with regard to the Entropy of v_k.

where $k^{(j)} = \text{argmax}_k \, I_n(z_j; v_k)$.

***Ad hoc* disentanglement measure:** The above measure assumes the tractability of its mutual information estimate, which assumes knowledge of the distribution of generative factors in the data at hand $p(v_k)$. This is a mild assumption when the target factor of variation is simple, such as a categorical factor. However, generative factors underlying textual content are often ill-defined and difficult to encapsulate in a simple probability distribution. In fact, disentangled textual representations must account for meaning, and therefore require estimating semantic similarity for evaluation which is still an open problem.

Textual disentanglement works often employ procedures that correlate representations with human semantic similarity judgement to report on semantic disentanglement. Two examples of such procedures can be found in the work of Huang et al. [2021a]. Namely, they measure correlation of embedding similarity with human similarity judgment, and train linear classifiers over their embeddings for paraphrase detection.

Alternatively, works employing Autoencoding models for disentanglement can leverage outputs from the decoder to quantify disentanglement. More precisely, for an embedding $z = \{z_1, \ldots, z_J\}$ corresponding to an observation x for which we want to study the relation between a generative factor of interest $v_k(x)$ and portion of interest z_p such that $p = \{j, \ldots, j'\} \subset J$, a common procedure employing the decoder consists in resampling z_p, either from a prior $p(z_p)$ or from $q_\phi(z_p|x')$, the encoding of a distinct sample x'. The rest of the representation $z_{\bar{p}}$ is then concatenated to the newly sampled ${z'}_p$ and decoded to produce a sample x_1. The analogous sampling and decoding procedure is also conducted in order to generate x_2 with the original z_p and a newly sampled ${z'}_{\bar{p}}$. Disentanglement is then studied by comparing $v_k(x)$ to $v_k(x_1)$ and $v_k(x_2)$. Given a similarity measure sim_{v_k} over realizations of the generative factor v_k, one concludes that v_k is disentangled in z and represented by z_p if, on average, $sim_{v_k}(v_k(x), v_k(x_2)) >> sim_{v_k}(v_k(x), v_k(x_1))$.

An instance of this *perturb-and-measure* procedure can be found in Xu et al. [2020a]. Conveniently this work provides 2 examples of similarity measures for assessing disentanglement: An exact match between the predictions of a pre-trained classifier for the sentiment factor, and a continuous score between 0 and 1 for content preservation (BLEU).

4.5 Applications in NLP

The main line of work in the area of disentanglement in NLP revolves around using multitask learning to separate concepts in neural representations. Works on unsupervised learning of disentangled representations remain rare compared to their supervised counterparts. Disentanglement without supervision on text data has first been explored in the work of Xu et al. [2020b] where they proposed a method to improve meaning preservation and applied it to separate sentiment or topic information from content in sentence representation. Other examples of works addressing unsupervised disentanglement on text data include that of Mercatali and Freitas [2021] who studied the use of discrete latent vari-

ables for such purpose, the work of Behjati and Henderson [2021] who worked on inducing morphemes-level representations using character-level Seq2Seq models, and Tjandra et al. [2021]'s work in which they disentangle *speech* content features from its style features by leveraging the intuition that content is local and style is global.

Works on *supervised* disentanglement in NLP are however abundant. Looking at their respective contribution types, these works can be divided in two categories: works with task-specific contributions and works which present contributions that aim to improve disentanglement in the general case.

Withing task-specific disentanglement, particular effort has been invested in separating semantics from form in general (meaning for example style) or syntax in particular. This is due to the fact that this separation proved useful for many applications, namely paraphrase detection [Chen et al., 2019a, Bao et al., 2019, Huang et al., 2021a] often accompanied by paraphrase generation [Romanov et al., 2019, Chen et al., 2019b, Zhang et al., 2019b, Huang and Chang, 2021a, Hosking and Lapata, 2021, Li et al., 2021a, Hosking et al., 2022] as well as for Machine Reading Comprehension [Wu et al., 2022]. Additionally, task-specific disentanglement has been used to enforce fairness by dissociating sensitive attributes from task relevant information in the prediction process [Colombo et al., 2022].

Concerning contribution that aim to improve disentanglement, methods generally fall within two categories: Methods that use Information Theoretic constraints to enforce dependence or independence between latent variables and generative factors [Cheng et al., 2020, Mercatali and Freitas, 2021, Colombo et al., 2021], and Methods that use adversarial learning schemes to enforce separation between information in latent variables [Romanov et al., 2019, John et al., 2019].

Concerning the types of inductive biases discussed in section 4.3, the vast majority of NLP works addressing disentanglement seem to employ supervised learning as inductive bias. Only a few attempts have been made at using unsupervised inductive bias, *i.e.* using known characteristics of language to direct information towards specific parts of latent representations. For instance Chen et al. [2019a], Li et al. [2021a] and Huang and Chang [2021a] discard word order information from semantic embedding modules in order to encourage syntactic embeddings to contain this information[5].

4.6 Conclusion

This chapter served laying the groundwork for a major component of the contributions described in this thesis: Disentanglement. In Section 4.1, after defining and motivating disentanglement, we explained how VAEs factor observations into independent generative factors, as dictated by the intuitive understanding of what disentanglement requires. To provide deeper understanding of how VAEs truly operate on data, and how that may relate to a set of target generative factors, we explained in Section 4.2 Locatello's impossibility

[5]This invariance of semantic representations to word order is, of course, not a viable design choice since it leads to sentences such as *"John loves Mary."* and *"Mary loves John."* to have the same semantic representations.

result regarding unsupervised disentanglement, and the relation put forward by Rolinek's work between VAEs and PCA.

Since VAEs *only* factor observations into independent latent factors, we explained in Section 4.3 alternative inductive biases used in the literature to induce disentanglement. Moving on to more practical notions, we summarized in Section 4.4 the different ways disentanglement can be measured and described MIG and MIG-sup to give examples, but also to provide better understanding of disentanglement metrics designed in this thesis (Chapter 8) which were inspired by MIG.

Finally, we provided in Section 4.5 an overview of the current NLP disentanglement landscape with an emphasis on two important observations:

- Unsupervised disentanglement in NLP is a research direction that needs more exploration.
- Most of the effort expanded towards disentanglement in NLP relies on labeled data, information theory-based or adversarial training-based objectives. In fact, little-to-no contributions in this area address the design of linguistically inspired built-in inductive bias for models to spontaneously induce (or help induce) disentanglement.

Chapter 5

Transformers and their NLP applications

Transformers have been introduced by Vaswani et al. [2017] as an architecture to process *sets* of observations in general, or *sequences* of observations in the particual case of linguistic observations. Contrary to RNNs, the *de facto* language modeling architecture before Transformers, this alternative architecture computes a contextualized representation of each element of the input sequence using a *parallelizable* computation method, as opposed to the recursive computations implemented within RNNs. The core component that enables this parallelizable contextualisation in Transformers is called *attention* and has been used in Deep Learning way before the introduction of Transformers [Schmidhuber, 1992, Bahdanau et al., 2015, Luong et al., 2015].

Besides parallelizability, the quick adoption of attention was driven by the clear meaning behind its computation: Assigning a *weight* to each element in a sequence in order to calibrate the extent to which it participates in the prediction at hand. Simply put, this mechanism calculates the degree to which *attention* must be paid to each element of the sequence, hence its name. On top of performance gains often brought by attention, its clear meaning enables inspecting attention weights to understand the rationales behind predictions at test time. An instance where this feature can prove crucial is, for example, that of Deep Learning systems subject to strict fairness regulation such as the attention-based hireability prediction system of Hemamou et al. [2019].

Transformers were widely adopted after the introduction of the first Transformer-based MLM: Bidirectional Encoder Representations from Transformers (BERT; Devlin et al., 2019). The ensuing prolific research on Transformer-based language modeling can be described along three Major research directions:

- *Pre-training and transfer*: This area pertains to self-supervised Transformer training schemes which yield models that perform well on tasks for which they are only fine-tuned with minimal data and computational resources.

- *Language Modeling*: This area looks into the use of Transformers for causal (classical) language modeling, and takes special interest in scaling them to unusually large corpora with an unusually large number of parameters.

- *Model Interpretability and Analysis*: This area looks into the inner-working of Transformer-based NLP systems with a special focus on BERT and its variants.

The disentanglement-related contributions we present in this thesis are all built upon Transformers attentions as an inductive bias. In that sense, we dedicate this chapter to providing in-depth explanations on attention and Transformers. We first describe the initial motivation behind attention, as well as the first formulations proposed to calculate it (§ 5.1). Moving on to the subject matter, we formally describe the Transformer architecture introduced by Vaswani et al. [2017] (§ 5.2). The subsequent section (§ 5.3) consists in an overview of Transformer-based language modeling where we present MLMs such as BERT and its variants(§ 5.3.1), prominent *causal* LMs and VAE-based Transformer LMs(§ 5.3.2), and a summary of findings from BERT-specific studies (a.k.a. *Bertology*) in particular and Transformer-based LM analysis in general (§ 5.3.3) .

5.1 The Core Component of Tranformers: Attention Mechanisms

As explained above, the purpose of an attention mechanism is to produce scores that are used to weight the contribution of each sequence element in a sequence-level representation. Formally, a sequence of token-level representations $h = \{h_1, \ldots, h_N\}$ can be aggregated to a sequence-level representation $\bar{h}$ using attention as follows:

$$\bar{h} = \sum_{i=1}^{N} h_i f_i(h) \tag{5.1}$$

$$\text{s.t.:} \quad f_i(h) = \text{softmax}_i(\{s(h_1, c), \ldots, s(h_N, c)\}) \tag{5.2}$$

$$= \frac{exp^{s(h_i,c)}}{\sum_j exp^{s(h_j,c)}} \tag{5.3}$$

where s is a scoring function that calculate the *unnormalized* weight of each sequence element, and c is a context element with regard to which attention is calculated. Across the literature, attention mechanisms mainly differ in the way s and c are formulated.

Attention before Transformers: In the pre-Transformer era, for applications such as sequence classification, c is was usually posited to be a vector (or multiple vectors) of constant learnable parameters [Yang et al., 2016]. The intuition behind this design choice was to calculate *relevance scores* with regard to the task at hand, which was constant and therefore represented by a fixed context vector c.

The more sophisticated uses of attention, which led to the Transformer architecture, arose from sequence-to-sequence prediction tasks such as Machine Translation [Bahdanau et al., 2015] or Speech Recognition [Chorowski et al., 2014]. In this setup, the informativeness of each element of the source sequence depends on the step reached in the target sequence generation. Accordingly, a context vector c_j is calculated at each generation step j as a function of the current state of the generator (or decoder). This use of attention in

the sequence-to-sequence setup was introduced by Bahdanau et al. [2015], and the benefit of it was twofold:

- Seeking the most relevant source token to generate the current target token is similar to predicting *alignments* between source and target tokens. Simultaneously learning to translate and align was actually the main motivation behind Bahdanau et al. [2015]'s pioneering use of attention for sequence-to-sequence transduction.
- Contrary to previous sequence-to-sequence models in the literature [Sutskever et al., 2014], attention mechanisms allow calculating a representation of the source sequence that accommodates information pertaining each decoding step. In fact, the performance of previous models which used a single sequence representation for all generation steps was know to deteriorate for long sequences [Cho et al., 2014a].

Concerning the scoring function s, Bahdanau et al. [2015] used a feed forward neural network with a tanh activation function on the concatenation of context c_j, representing the current target generation step, and h_i which is a representation of the i^{th} source token:

$$s(h_i, c_j) = \tanh(W_s . \text{Cat}(h_i, c_j) + b_s) \tag{5.4}$$

where W_s and b_s are respectively a matrix and a bias term with learnable parameters and Cat is the concatenation operator. This formulation of attention later came to be known as *additive* attention. It was referred to as such in contrast to *multiplicative* formulation of the score function developed by Luong et al. [2015]. In this formulation attention is rather computed as a dot product between the current target context, and a linear transformation of the source token representation:

$$s(h_i, c_j) = c_j^T \cdot W_s \cdot h_i \tag{5.5}$$

where W_s is a matrix with learnable parameters.

As mentioned above, Bahdanau et al. [2015]'s use of attention was, in part, an attempt at learning alignments between source and target words in Machine Translation without using annotations on alignment. Their work, as well as subsequent works using attention, was successful at inducing such alignments and led to results such as the ones depicted in Figure 5.1.

Quantifying the influence of source tokens on target tokens using attention, as is done in Figure 5.1, is important technique which serves as a basis for quantitative evidence presented later in Chapter 8.

Attention within Transformers: Transformers attention is multiplicative attention augmented with a few changes. A first change, we deem important, is a change in terminology. Although it was hinted prior to Transformer, that attention is a "*soft*-search"

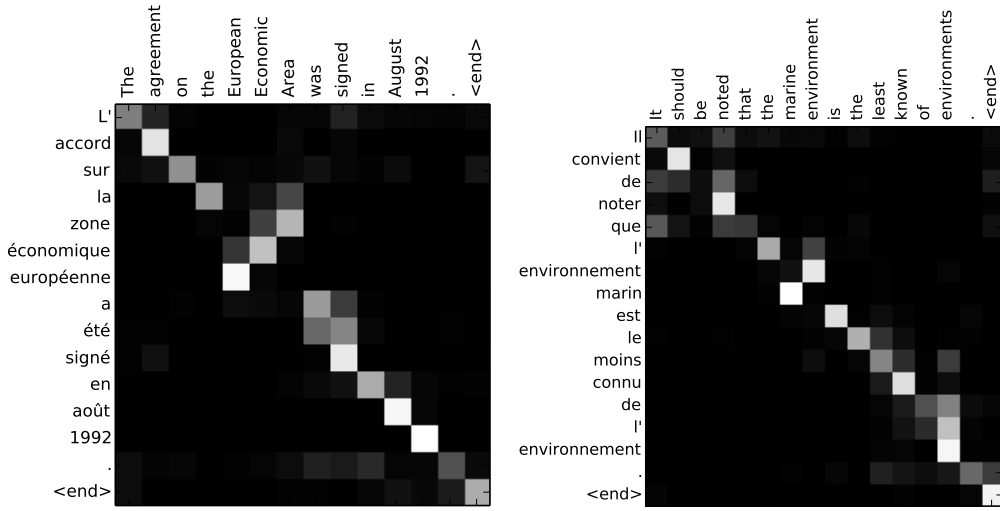


Figure 5.1: Two example alignments between source and target tokens induced by attention values in the work of Bahdanau et al. [2015]. The brightness of each cell is proportional to the value of $s(h_i, c_j)$, where the columns i correspond to source tokens and the rows j correspond to target tokens.

module [Bahdanau et al., 2015], this interpretation of its functioning scheme is explicited in Transformers terminology. In fact, source hidden states h_i are said to formulate *keys* (K) and *values* (V), and target hidden states, previously referred to as contexts, are said to formulate *queries* (Q) that call for the source hidden states according to their keys, and aggregate their values using the obtained weights. Formally, for a given query Q_j and a set of source keys $K = \{K_1, \ldots, K_N\}$ and values $V = \{V_1, \ldots, V_N\}$, the result of attention for the target with index j is formulated as follows:

$$\text{Attention}(Q_j, K, V) = \sum_{i=1}^{N} f_i(Q_j, K) V_i \tag{5.6}$$

$$\text{s.t. :} \quad f_i(Q_j, K) = \text{softmax}_i(\frac{Q_j \cdot K^T}{\sqrt{d}}) \tag{5.7}$$

where d is the size of the representations Q_j and K_i. Normalizing by the size of the representation is one of the changes brought by Transformers attention, and benefits the model with more numerical stability according to Vaswani et al. [2017]. When considering the full range of targets $Q = \{Q_1, \ldots, Q_{|L|}\}$, Transformers attention can be abbreviated in its efficient matrix multiplication formulation as follows:

$$\text{Attention}(Q, K, V) = softmax(\frac{Q.K^T}{\sqrt{d}}).V \tag{5.8}$$

In the above, we insist on the importance of the change in terminology as it constitutes the basis of the intuition behind the last work described in this thesis (Chapter 9).

The last difference between Transformers attention and standard multiplicative attention is called *Multi-Head Attention* (MHA). Specifically, in contrast to previous Neural

Networks employing attention, Transformers utilize a multitude of attention mechanisms in parallel and concatenate their results:

$$\text{MHA}(\tilde{Q}, \tilde{K}, \tilde{V}) = \text{Cat}(head_1, ...head_H)W^O \tag{5.9}$$

$$\text{s.t} : head_i = \text{Attention}(\tilde{Q}W_i^Q, \tilde{K}W_i^K, \tilde{V}W_i^V) \tag{5.10}$$

where W_i^Q, W_i^K, W_i^V, and W^O are matrices with trainable parameters.

5.2 Transformer Architecture

The Transformer architecture [Vaswani et al., 2017] has been proposed for sequence-to-sequence processing, and therefore consists of a sequence encoder and a sequence decoder.

Throughout encoding and decoding, each MHA layer and each feed-forward layer are followed by a residual connection [He et al., 2016a], which is crucial for scaling to deeper networks, and a Layer Normalization [Ba et al., 2016] which helps improve training stability and speed.

Transformer encoders: A Transformer encoder uses only one sequence of input elements S, which is typically a source language sequence of tokens in Machine Translation. Attention is used in this encoder for the different elements of the input sequence to exchange information; *i.e.* for contextualization. Specifically, Multi-Head Attention internally calculates its queries, keys, and values (*cf.* Eq. 5.10) from this single input sequence, and thus produces contextualized sequence elements. This single input Multi-Head Attention is called Self-Attention (SA):

$$\text{SA}(S) = \text{MHA}(S, S, S) \tag{5.11}$$

Given this operator, a Transformer encoder with D^{enc} layers can be formulated as follows[1]:

$$\text{Enc}(S) = \tilde{S}_{D^{enc}}, \text{ s.t. } \tilde{S}_d = \begin{cases} S \text{ if } d = 0 \\ \text{F}_d(\text{SA}_d(\tilde{S}_{d-1})) \text{ if } d > 0 \end{cases}$$

where each F_d is a feed-forward neural network. Feed-forward neural networks in Transformers are formulated as two linear transformations with a Rectified Linear Unit (ReLU) activation in between[2]:

[1] Here, we omit layer normalization and residual connections to lighten these equations, although they are essential to learn Transformers reliably [He et al., 2016a, Ba et al., 2016].

[2] The ReLU activation is the function $f : x \longrightarrow \max(0, x)$. Subsequent Transformer-based models such as GPT [Radford et al., 2018] and BERT [Devlin et al., 2019] replaced this function by a Gaussian Error Linear Unit (GELU) activation function, which is a continuous version of ReLU introduced by Hendrycks and Gimpel [2016] who show that it empirically outperforms ReLU across various NLP tasks, among others.

$$\mathrm{F}(x) = \max(0, x.W_1 + b_1).W_2 + b_2 \tag{5.12}$$

where W_1 and W_2 are learnable parameter matrices, and b_1 and b_2 are learnable bias terms.

Transformer decoders: A transformer decoder takes as an input a source sequence of elements S, and a target sequence of elements T. In the decoder, attention is used in two manners. The first is an SA module that exchanges information between elements in the target sequence. The second is a *Cross-Attention* (CA) module which *pulls* information from source *values* through their *keys* with target *queries*:

$$\mathrm{CA}(T, S) = \mathrm{MHA}(T, S, S) \tag{5.13}$$

A Transformer decoder with D^{dec} layers can be formulated as follows[3]:

$$\mathrm{Dec}(T, S) = \tilde{T}_{D^{dec}}, \text{s.t. } :$$

$$\tilde{T}_d = \begin{cases} T \text{ if } d = 0 \\ \mathrm{F}(\mathrm{CA}(\mathrm{SA}(\tilde{T}_{d-1}), S)) \text{ if } d > 0 \end{cases}$$

NLP-specific processing in Transformers:[4] To process textual sequences with Transformers, textual tokens must be embedded and their positional information must be encoded withing their embedding vectors. The token embedding step simply employs an embedding lookup layer, which is a standard component in Deep Learning-based NLP. However, contrary to previous sequence processing architectures such as LSTMs [Hochreiter and Schmidhuber, 1997], Transformers are order-invariant, and therefore require engineering a mechanism to encode word order information. Vaswani et al. [2017] chose a mechanism, which they called *positional encoding*[5], that consists in injecting order information by adding the amplitudes of *sine* and *cosine* functions applied to different

[3] Layer Normalization and Residual connections are also omitted here.

[4] We refer to the implementation details described here as *NLP-specific* since they were originally engineered for textual inputs. Nevertheless, they remain applicable to any discrete times series, and the positional encoding, described later in this section, remain applicable to any type of ordered input sets (*e.g.* image patches; Lu et al., 2019b).

[5] This is neither the first, nor the best attempt at encoding positions. For instance, Gehring et al. [2017] train a separate embedding for each position while experimenting with attention-augmented Convolutional Neural Networks for sequence-to-sequence modeling, and Shaw et al. [2018] show that embedding relative, as opposed to absolute, positions improves results for Machine Translation.

position-dependant frequencies:

$$\text{PE}_{pos,2i} = \sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{5.14}$$

$$\text{PE}_{pos,2i+1} = \cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{5.15}$$

where *pos* is the position of the token, and i is the dimension at which addition is being performed in the token embedding vector. This positional encoding operation is applied to both source and target input tokens. The intuition behind Vaswani et al. [2017] choosing sinusoidal functions is that the model should better attend to relative positions since, for any fixed offset k, PE_{pos+k} is a linear function of PE_{pos}.

To model textual sequences in an auto-regressive manner, Transformers are also required to restrict the information exchange between target tokens so that target sequence elements only pull information from previous tokens. In practice, this is enforced through a binary mask M on attention, leading to what is called *masked attention*:

$$\text{Attention}(Q, K, V) = (\text{softmax}(QK^T) \odot M)V \tag{5.16}$$

$$\text{s.t. :} \quad M_{ij} = \begin{cases} 1 \text{ if } i \leq j \\ 0 \text{ if } i > j \end{cases} \tag{5.17}$$

This masked attention is typically used in Transformer decoders for SA between the target elements.

Given the above adaptations, NLP-specific Transformers only further require a linear layer with softmax activations in order to produce word probabilities for sequence modeling.

The Transformer architecture for textual sequence-to-sequence modeling: Figure 5.2 displays how the Transformer encoder and decoder are assembled with NLP-specific processing steps for the textual sequence-to-sequence modeling in the work of Vaswani et al. [2017].

5.3 Language Modeling with Transformers

In this section, we go over recent Transformer-based trends in NLP research, namely through the 3 axes along which they can be categorized: masked language modeling and pre-training, causal language modeling, and LM Analysis.

5.3.1 Masked Language Modeling and Pre-training with Transformers

The flagship of MLMs is the very first model to use masked language modeling: BERT [Devlin et al., 2019]. This model encodes tokens using only the Transformer encoder from the architecture depicted in Figure 5.2. BERT is trained to encode consecutive sentences $A = \{a_1, \ldots, a_{|A|}\}$ and $B = \{b_1, \ldots, b_{|B|}\}$ tokenized with a WordPiece Tokenizer [Schuster

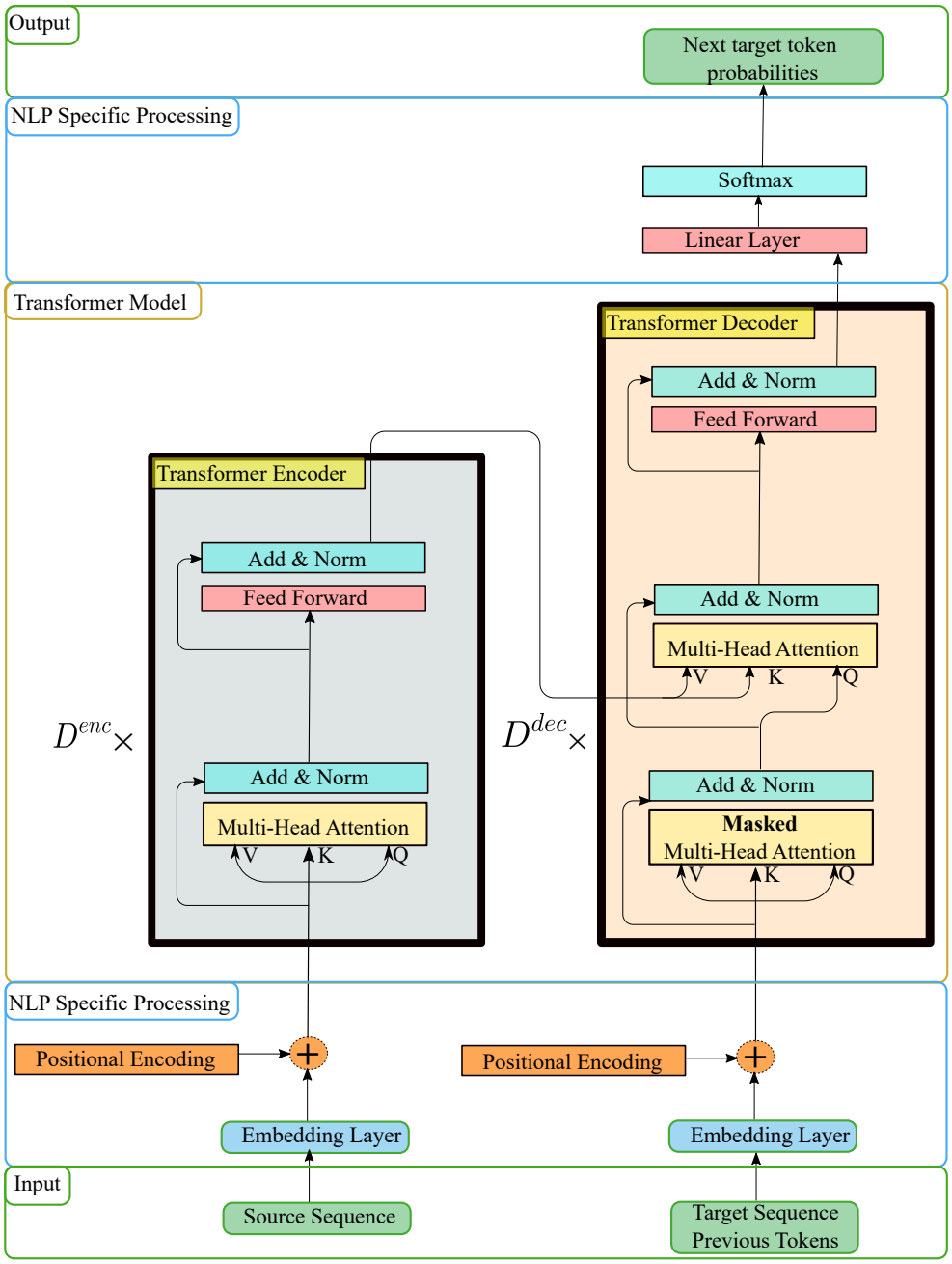


Figure 5.2: The Transformer architecture for textual sequence to sequence modeling. The encoder and decoder are recursively applied D_{enc} and D_{dec} times respectively. As emphasized in the figure, the middle part *"Transformer Model"* constitutes the core Transformer architecture, which is input agnostic, while the parts above and below this core part represent NLP-specific adaptations.

and Nakajima, 2012]. These sentences are provided to the model under the form $\{[CLS], a_1, \ldots, a_{|A|}, [SEP], b_1, \ldots, b_{|B|}\}$[6] where $[CLS]$ and $[SEP]$ are special tokens. Apart from summing token embeddings with their positional encodings, BERT also sums them with a an additional segment embedding, where the segment is either A or B.

Pre-training BERT consists in optimizing it for two tasks on 16 GB of text data from

[6]Notice that BERT does not use and end-of-sentence token, since it does not perform language generation.

BookCorpus [Zhu et al., 2015] and english Wikipedia. The first task is masked language modeling as described in Section 2.5 (Eq. 2.16). The second is a Next Sentence Prediction (NSP) task, where C, the output representation of the $[CLS]$ token, is fed to a binary classifier which predicts whether sentence B is the sentence that follows A in the corpus, or a randomly sampled sentence from this corpus.

After pre-training, BERT is *finetuned* on tasks from the General Language Understanding Evaluation suite (GLUE; Wang et al., 2018). BERT outperformed the previous state-of-the-art on the GLUE benchmark by a large margin, which opened an avenue for research on Transformer-based Pre-trained LMs.

Numerous works after BERT succeeded in establishing new state-of-the-art performance on language understanding using variants of its pre-training scheme. Yang et al. [2019] use a Transformer architecture optimized for longer sequences [Dai et al., 2020], and reformulate the masked language modeling objective as an auto-regressive language modeling objective that goes through tokens in a randomized order. Liu et al. [2020] find that training BERT longer, with larger batches, more data (160 Gb) and without the NSP objective substantially improves performance on downstream tasks.

Besides the above works, a number of studies explored modifying BERT while aiming for improvements other than better downstream task transfer. For instance, ALBERT [Lan et al., 2020] outperforms BERT on language understanding benchmarks with less parameters by using a single layer that applies to the inputs multiple times in order to simulate a multi-layer Transformer. Also on the parameter efficient side, Sanh et al. [2019] distill the original BERT to 60% of its original size while retaining 97% of its original performance. To improve training speed, Clark et al. [2020] train a Transformer encoder (ELECTRA) to distinguish real tokens from fake tokens, rather than to generate tokens in masked placeholders, and achieve performance similar to that of Liu et al. [2020]'s approach while using only $\sim 22\%$ of their compute[7].

Pre-trained Transformers have also been proven effective outside of BERT's *token embedding* paradigm, namely in the sequence-to-sequence setup [Raffel et al., 2020, Lewis et al., 2020], and the multimodal (language and vision) setup [Lu et al., 2019a, Li et al., 2021b, Huang et al., 2021c].

5.3.2 Causal Language Modeling with Transformers

In addition to lower perplexity scores, the ability of Transformers to scale to high volumes of data benefited causal language modeling in that it enabled verifying a crucially important hypothesis for the future of AI as a whole that we discuss in what follows: "*Language models are unsupervised multitask learners*" [Radford et al., 2019][8].

Through the relatively large-scale[9] LM GPT-2, Radford et al. [2019] showed that language modeling is a task that involves learning, and therefore requires solving, multiple

[7] Compute, here, was measured in Floating Point Operations (FLOPs).
[8] The quoted sentence is actually the title of Radford's paper
[9] GPT-2 used 40Gb of text for its training which, at the time, was considered "large-scale".

non-trivial reasoning subtasks for which there are no labels in the data apart from the text itself, hence the *unsupervised multitask learning* mentioned by Radford. For instance, a large-scale LM is likely to encounter many times the phrase "TL;DR"[10], which is a cue signaling the presence of a small summary. Consequently, one can probe the ability of an LM to summarize by prompting it with a text appended with the phrase "TL;DR", and scoring the completion generated by the LM with regard to a gold standard summary. Similar prompting strategies can be developed for a number of tasks like translation (*e.g. <French sentence>*'s translation in English is *<LM completion>*), reading comprehension (often investigated with conversational prompts), or even basic arithmetics (*e.g. <x> times <y> is <LM completion>*). Radford et al. [2019] succeed in showing that LMs learn these tasks underlying language modeling by demonstrating the ability of GPT-2 to score at least significantly above random chance at the aforementioned tasks.

The recent years have seen the emergence of Transformer-based LMs larger than GPT-2 by multiple orders of magnitude. The first instance of such models, which was GPT-3 [Brown et al., 2020], consisted of 175 billion parameters (approximately 116 times the size of GPT-2) and was trained on 540 Gb of data (approximately 11 times the data used for GPT-2). GPT3 demonstrated strong few-shot learning performance through pure prompting on a wide range of tasks, and even generated short news articles (∼200 words) which were nearly unidentifiable[11] by humans. Extremely large LMs subsequently started emerging as a research direction primarily fueled by industrial fundings [Rae et al., 2021, Chowdhery et al., 2022, Smith et al., 2022, Du et al., 2022, Thoppilan et al., 2022] with a specific focus on how they can solve tasks through prompting with few-to-no training steps [Su et al., 2022].

Due to the difficulties incurred by posterior collapse, Transformers have rarely been trained as VAEs for language modeling, and latent variable LMs have therefore been relatively left out of the large-scale language modeling race. To the best of our knowledge, Optimus [Li et al., 2020b] and DELLA [Hu et al., 2022] constitue the only attempts at building large-scale VAE LMs. Optimus uses BERT as an encoder and GPT-2 as a decoder to demonstrate that Transformer-based VAE LMs can outperform their standard Transformer-based LM counterparts when similar effort is expanded to train them. But seeing that Optimus still suffers from posterior collapse to a certain extent, Hu et al. [2022] proposes through DELLA a hierarchical latent variable modeling scheme which they prove to be effective in dealing with posterior collapse to a large extent. Given that DELLA is very recent, its potential effect on the language modeling landscape is yet to be seen.

[10]This phrase is internet lingo for "Too Long; Didn't Read". Although the phrase was originally meant to be commented on *long* posts, it came to be used as a marker for small abstracts accompanying long internet posts that may be skipped by readers due to their length.

[11]When asked to identify the model-generated article between a GPT-3 article and a human-generated article, human operators had a 52% accuracy with a confidence interval of 49%-54%. Random chance accuracy (50%) being in this interval, the experiment could not conclude that humans were capable of discerning GPT-3 articles from human articles.

5.3.3 Bertology and Transformer-Based Model Analysis

The previous sections enumerated the successes Transformers have had across various NLP tasks, which necessarily raises the question: How do they do it ? A flurry of studies, following the pioneering work of Devlin et al. [2019], have dived into the inner working of Transformers in order to make sense out of the information flow in this architecture, and how it relates to what we know about linguistics. A digest of the BERT-specific portion of these works (a.k.a *Bertology*) can be found in Rogers et al. [2020b].

The first series of works on the subject tackled identifying the content of each layer through *probing*, a technique that consists in training a classifier on top of a representation and measuring the performance of the classifier with the intuition that the higher it is, the more informative the representation is about the target concept. When applying this methodology to BERT, Tenney et al. [2020] observed that it tends to process information in a manner that bears resemblance to the classical NLP pipeline, meaning that the first layers tend to encode low-level surface form information such as PoS Tags, the middle layers appear to be informative on more advanced syntactic relations such as dependencies, while semantic information is either in the final layers (*e.g.* for co-reference resolution) or spread out throughout the entire network as is the case for relation classification. These findings were corroborated by later studies on BERT [Jawahar et al., 2019b, Hewitt and Manning, 2019b], but do not seem to generalize to all BERT-like models. As a matter of fact, Fayyaz et al. [2021] have shown that linguistic information emerges earlier compared to BERT in the layers of XL-Net [Yang et al., 2019], and later in the layers of ELECTRA [Clark et al., 2020].

In order to better understand the way Transformers encode linguistic information, a number of works have focused on the extent to which these models grasp syntactic notions. To that end, Marvin and Linzen [2020] developed a test suite with minimally modified pairs designed to assess specific syntactic capabilities in LMs such as subject/verb agreements and correct selection of negative polarity items. By compiling this evaluation suite together with similar syntactic and psycholinguitic evaluation toolsets [Wilcox et al., 2018, Futrell et al., 2018, Wilcox et al., 2019], Hu et al. [2020] compared a range of popular architectures for language modeling and observed that the syntactic capabilites of the different models they tested *was not* correlated with their respective perplexities, meaning that advances in language modeling measured by perplexity do not translate to models that better understand the syntax underlying their training corpora. Additionally, their experiments revealed that syntactic capabilities were more influenced by architectures than they were by the size of the training corpora, with the Transformer architecture tested with GPT-2 coming on top of previous state-of-the art RNN-based architectures such as RNNG [Dyer et al., 2016] and ON-LSTM [Shen et al., 2019]. Subsequent studies on the presence of syntactic information in Transformers have backed and better justified the ability of Transformers to process syntactic information by showing that Transformers encode information in tree-like structures [Jawahar et al., 2019b, Hewitt and Manning,

2019b].

Concerning the role played by different Transformer components in formulating output representations, Geva et al. [2021] proposed and leveraged an interesting perspective on feed-forward layers in Transformers to provide a unified view on information processing in Transformers: the two matrices forming these layers (W_1 and W_2 in Eq. 5.12) function as keys and values where rows from the first matrix are used to weight (or call) lines from the second matrix. Using this insight they show that the values called by keys in these matrices correspond to human-understandable textual patterns. Then, In Geva et al. [2022], the same authors show that these feed-forward layers operate on outputs by gradually promoting concepts, which are also largely understandable, in their outputs while forming the word representations. Also pertaining to the association between Transformer components and understandable concepts, Clark et al. [2019] have shown that many attention heads in BERT emulate identifiable syntactic functions such as dependency relations and co-references with high accuracy. They also show that BERT attention heads learn to default to special tokens ([CLS] and [SEP]) when they can't find the target argument for their specific linguistic functions.

It is worth noting that, alongside a better understanding of Transformers, this line of works also led to numerous advances in methodology concerning, for instance, the use of attention as an explanation [Jain and Wallace, 2019, Wiegreffe and Pinter, 2020], the validity of probing [Pimentel et al., 2020a], or contrastive evaluation with minimal pairs [Kodner and Gupta, 2020, Vamvas and Sennrich, 2021].

5.4 Conclusion

This background chapter is aimed at explaining crucial architectural components for our contributions on disentanglement: Transformers and attention. We first motivated attention mechanisms and introduced the first attempts at formulating them (additive and multiplicative), as well as the first alignment results they exhibited in Machine Translationng (§ 5.1). We then showed in the same section how Transformers normalized multiplicative attention, and duplicated it into Multi-Head Attention to form their basic building block. Given this attention mechanism, we explained in section 5.2 how Transformers exchange information between sequence elements (*i.e.* Self-Attention) or pull information from source sequences to target sequences (*i.e* Cross-Attention). In the same section, we also explained how these attention blocks, together with other Transformer components come together to form the complete multi-layer Transformer architecture, and how NLP-specific measures, such as positional encoding and masked attention, must be applied in order to deal with textual inputs and outputs.

Moving to their applications, we painted in Section 5.3 a picture of the current landscape of Transformer-based works in NLP. Specifically, we descibed the first MLM, BERT, and gave a summary of the subsequent models that modify and improve upon it (§ 5.3.1). In the following subsection (§ 5.3.2), we outlined works on large-scale language modeling

with Transformers and how they exhibited the ability of LMs to solve a range of reasoning tasks underlying simple text completion. We also provide a summary of the few works attempting large-scale VAE-based language modeling with Transformers. Finally, we give an overview of the different conclusions drawn by works attempting to understand the inner-working of Transformer-based LMs (§ 5.3.3).

Chapter 6

Syntactic Structure of Sentences

The approach to interpretable machine learning in the context of NLP is peculiar in that language, the input data, is a media that has been subjected to meticulous scrutiny for millennia[1]. The myriad works investigating and theorizing about language provide solid foundations for interpretable NLP to lean on. As briefly discussed in the previous chapter (§ 5.3.3), works analyzing the behavior of Transformers are largely built on relations to linguistic notions [Hewitt and Manning, 2019b, Tenney et al., 2020, Hu et al., 2020]. Similarly, we build in this thesis interpretable representation learning techniques for NLP which are largely motivated by their meaningfulness with regard to linguistics in general, and *syntax* in particular.

Syntax is the area of linguistics that looks into the way words combine in order to produce phrases or sentence that abide by the grammar of a language. Although theoretical frameworks formalizing this set of rules may vary, it is generally agreed upon that words in a sentence can be embedded in a tree-like structure describing its syntax, where these words combine into phrases. Figure 6.3[2] is a minimal example illustrating this combination process through the dependency tree of a sentence.

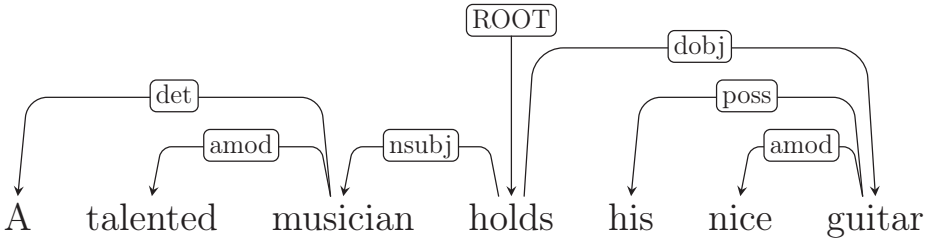


Figure 6.1: A sentence and its dependency structure. The structure is built as a tree over words in the sentence, where arrows go from *heads* (or *parents*) to *dependents* (or *children*).

Beyond enumerating valid constructions in a certain language, syntax reflects the way minimal semantic units are *composed* to convey complex meaning. For instance, depen-

[1]Dependency grammars for instance, approached in Section 6.2, have first been formulated by the Indian scholar Panini around the 5^{th} century B.C.E. [Joshi et al., 1991].

[2]These dependency relations can be obtained using Spacy's online dependency parser which can be used at this URL: https://explosion.ai/demos/displacy

dency relations illustrate the compositional nature of language and the role of syntax in this composition by relating words through *head-to-dependent* relations. These relations model the way words in sentences *feed* off the meaning of other words in a hierarchical modification process that yields the overall meaning of the composed phrase or sentence [Tesnière, 1959, Chapter 21]. With that in mind, recall that the core component of Transformers, discussed in the previous chapter (§ 5.1), is attention: a mechanism that organizes the exchange of information between token representations, *i.e.* the way representations are *composed*. Understanding the relation between the way attention behaves and the linguistic theory describing the rules for phrase and sentence composition, *i.e.* syntax, is therefore crucial in making sense of NLP systems in the Transformers era.

In light of the above, we dedicate this chapter to presenting the syntactic notions which inspired our contributions, and provided grounding for the evaluation protocols used to validate them. First, we introduce *constituency analysis* (§ 6.1) and its backbone, Context-Free Grammars(CFGs). Subsequently, we present an alternative to the constituency view on syntax: *dependency analysis* (§ 6.2). This second section also emphasizes the strong relation of dependencies to semantics and predicate-argument structures.

6.1 Constituency Analysis

The constituents, referred to in *constituency analysis*, are groups of words that can behave as a single unit(*cf.* Chomsky, 1957, Chapter 4 and Jurafsky and Martin, 2022, Chapter 12). In other words, constituency analysis is a description of:

- the way words form constituents, *i.e.* multi-word units.
- the *type* of each of these constituents.

Classifying constituents into different types is useful for describing rules governing word grouping, *e.g.* A *determiner* (DT) and a *noun* (NN) form a *Nominal Phrase* (NP) in English, which in turn may be followed by a *Verb Phrase* (VP) to form a sentence (S). The set of such *production rules* for a certain language can be formalized through what is called a Context-Free Grammar[3] (CFG). For instance, the aforementioned production rules for English would appear in a CFG as follows:

$$NP \longrightarrow DT\ NN \tag{6.1}$$

$$S \longrightarrow NP\ VP \tag{6.2}$$

The first line from the above rules means "a nominal phrase may be formed by a determiner and a noun (in that order)". Symbols used in production rules belong to two categories: *terminal* and *non-terminal* symbols. terminal symbols are the words that actually appear in sentences (*e.g.* car, he, draw, ...) and non-terminal symbols are the abstractions

[3]Note that CFGs are simplified models for the syntax of natural languages. In fact, as shown by Shieber [1985], some syntactic phenomena exhibited by natural languages fall outside of the range of CFGs.

(or types) that may produce words or other non-terminals (*e.g.* noun, preposition, verb phrase, ...). The subset of non-terminals that produce terminals, such as *determiners* (DT) and nouns (NN), are called Parts-of-Speech (PoS). With the rules defined over terminal and non-terminal nodes, the set of possible strings of terminals that can be generated by a CFG from the starting symbol S forms a formal language. Conversely, sentences that can not be arrived at from the rules within a CFG are said to be *ungrammatical* with regard to that CFG.

Given these rules, constituency trees are constructions over sentences that depart from the starting symbol node S and use production rules from the CFG of the language at hand to arrive at the leaf nodes containing the terminal symbols in the sentence. An example of such construction can be seen in Figure 6.2[4].

6.2 Dependency Analysis

As discussed in the introduction, dependency analysis illustrates sentence structure through syntactic relations between words [Mel'cuk et al., 1988]. These relations form a tree where each node is a word, in contrast to the constituency tree where words are only leaf-nodes.

[4]This constituency tree was obtained through AllenAI's online NLP demo platform available at this URL: https://demo.allennlp.org/constituency-parsing

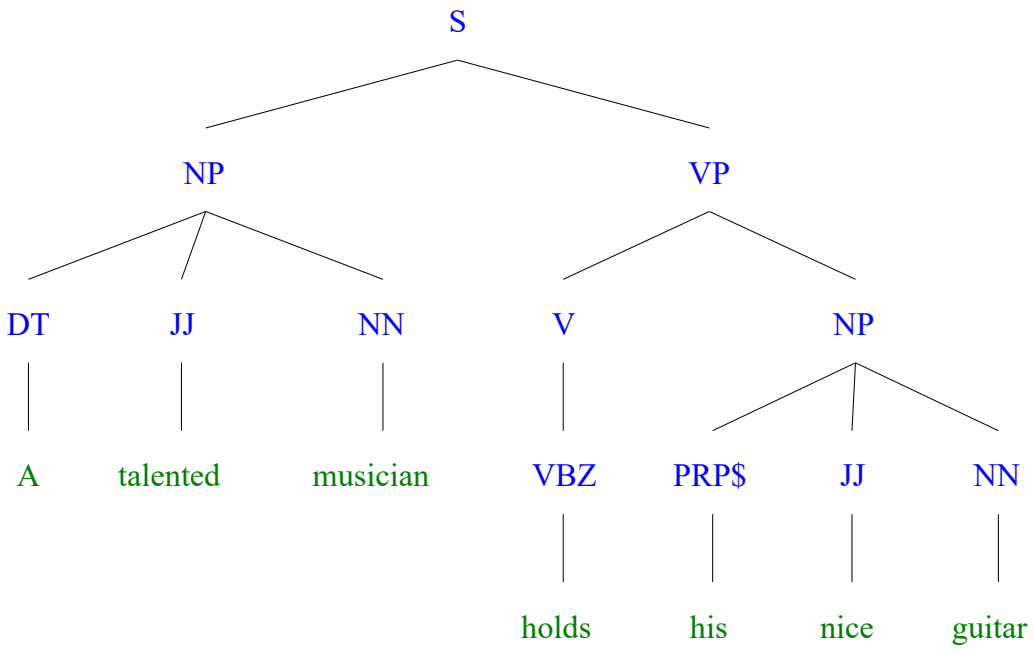


Figure 6.2: Example constituency tree for the sentence "A talented musician holds his nice guitar". Terminal nodes are labeled in green while non-terminal nodes are labeled in blue. JJ, VBZ and PRP are respectively "adjective, numeral or ordinal", "verb, present tense, 3rd person singular" and "pronoun, personal".

The words forming subtrees in a dependency tree (*i.e.* a node and all its descendants) correspond to words that form sentence constituents from the constituency perspective [de Marneffe et al., 2006]. However, the dependency perspective offers different information on this constituent. First, in a dependency tree we label arcs (relations between words) rather than nodes (constituent types). Second, a dependency subtree features a *head* word that is targeted by the syntactic functions of its direct children in the subtree; *e.g.* in "talented musician", "talented" is an *adjectival modifier (amod)* targeting the word "musician". Third, dependency trees are less rigid when it comes to word order[5]. Figure 6.3[6] shows the same dependency tree displayed in the introduction augmented with PoS tags, and semantic annotation concerning its predicative structure which is explicited below.

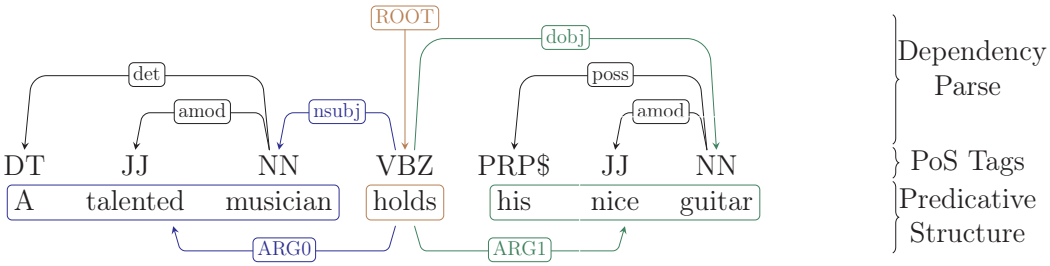


Figure 6.3: A sentence and its syntactic roles. The correspondence between syntactic roles and elements of the predicative structure is highlighted with colors. *ROOT* is the root node of the dependency tree, *nsubj* is a nominal subject, *dobj* is a direct object,*det* is a determiner, *poss* is a possessive determiner, and *amod* is an adjectival modifier. *ARG*0 and *ARG*1 are semantic *proto-roles* [Bonial et al., 2012] designating respectively the entity that performs the action (a.k.a the *agent*) and the entity affected by the action (a.k.a the *patient*).

The dependency structure of a sentence is especially interesting in that it strongly relates to the semantic structure of the sentence. In fact, formal theories of semantics, such as Generative Lexicon Theory [Pustejovsky, 1998], often posit that sentences translate to *lambda expressions*[7] which describe semantic composition as function applications over some abstract arguments. The lambda expression corresponding to the semantic processing underlying the sentence in Figure 6.3 can be written as follows[8]:

$$\lambda x.\lambda y.[musician(x) \wedge talented(x) \wedge nice(y) \wedge guitar(y) \wedge holds(x,y)] \tag{6.3}$$

This expression reads *"Let x and y such that x verifies musician, x verifies talented, y verifies nice, y verifies guitar, and (x,y) verifies holds"*. Notice that, in Figure 6.3, the part of the sentence characterizing the argument x corresponds to the *nsubj* dependency subtree, while the *dobj* subtree contains all the information about the argument y. The

[5]This property is interesting for languages with a *free word order* since dependency trees do not need to register different rules for different arrangements of the same syntactic roles.

[6]The semantic roles describing the predicative structure here can be obtained through AllenAI's online semantic role labeling demo available at this URL: https://demo.allennlp.org/semantic-role-labeling

[7]Refer to Rojas [2015] for an introdution to lambda calculus.

[8]The possession information expressed by the pronoun "his" is ignored here for simplicity.

ROOT node which is directly above *dobj* and *nsubj* is anchored to the verb *holds*, which informs about the main predicate expressed by the sentence. Identifying elements of the predicative structure, most often called *semantic role labeling*, is an NLP task introduced by Gildea and Jurafsky [2002] where portions of the sentence are labeled according to *semantic roles* which identify arguments of the semantic structure of the sentence. Although semantic role labeling started out with a large vocabulary of fine-grained semantic roles, subsequent annotation guidelines such as those of PropBank [Bonial et al., 2012] collapse semantic roles into a minimalistic set of *proto-roles* such as the ones used in Figure 6.3.

The correspondence of high-level syntactic roles such as *ROOT*, *nsubj*, and *dobj* with the main predicate and its arguments illustrated in Figure 6.3 is a well-marked pattern. As a matter of fact, statistics on the coNLL 2008 shared task on joint parsing of syntactic and semantic dependencies [Surdeanu et al., 2008] calculated by Lang and Lapata [2010] show, for instance, that 84% of constituents having the *agent* (ARG-0) semantic role have the *subject* syntactic role, and that 58% of constituents having the *patient* semantic role have the *object* syntactic role. This correspondance is strong enough for it to be emphasized by dependency annotation guidelines such as Universal Dependencies [Nivre et al., 2020] through a distinction between *core* syntactic roles, *i.e.* syntactic roles which directly relate to an element of the predicative structure such as the ones we color in Figure 6.3, and *non-core* syntactic roles[9].

The distinction between *core* and *non-core* syntactic roles, as well as the fact that core syntactic roles are placeholders segmenting the main information carried by the sentence are of particular importance to this thesis as they constitute the basis for the intuition behind the work presented in Chapter 8.

6.3 Conclusion

In this chapter, we introduced two syntactic analysis schemes, namely constituencies and dependencies. We explained that constituencies find their roots in CFGs, and describe sentences as the result of the successive application of the rules registered in the generative grammar describing the language at hand (§ 6.1). We use them to quantify the effectiveness of the method described in Chapter 9 of this thesis. More precisely, we study the degree to which syntactic information is present in latent variables where syntactic information ranges from shallow constituency structures, *i.e.* constituency trees cut at the 2^{nd} or 3^{rd} level, to deep constituency structures, *i.e.* entire trees. Example cuts of a constituency tree, also called syntactic templates, are displayed in Figure 6.4.

Concerning dependencies, we explained that they offer a perspective on sentences which explicits the role played by each word in the sentence structure and therefore better relates it to the manner in which it participates in building the meaning behind the sentence (§ 6.2). A particular notion that was emphasized in the section pertaining to dependencies is the relation between core syntactic roles and elements of the predicative

[9]As an example, Universal Dependencies provide a comprehensive inventory of syntactic roles they use with definitions and examples at the following URL: https://universaldependencies.org/u/dep/index.html

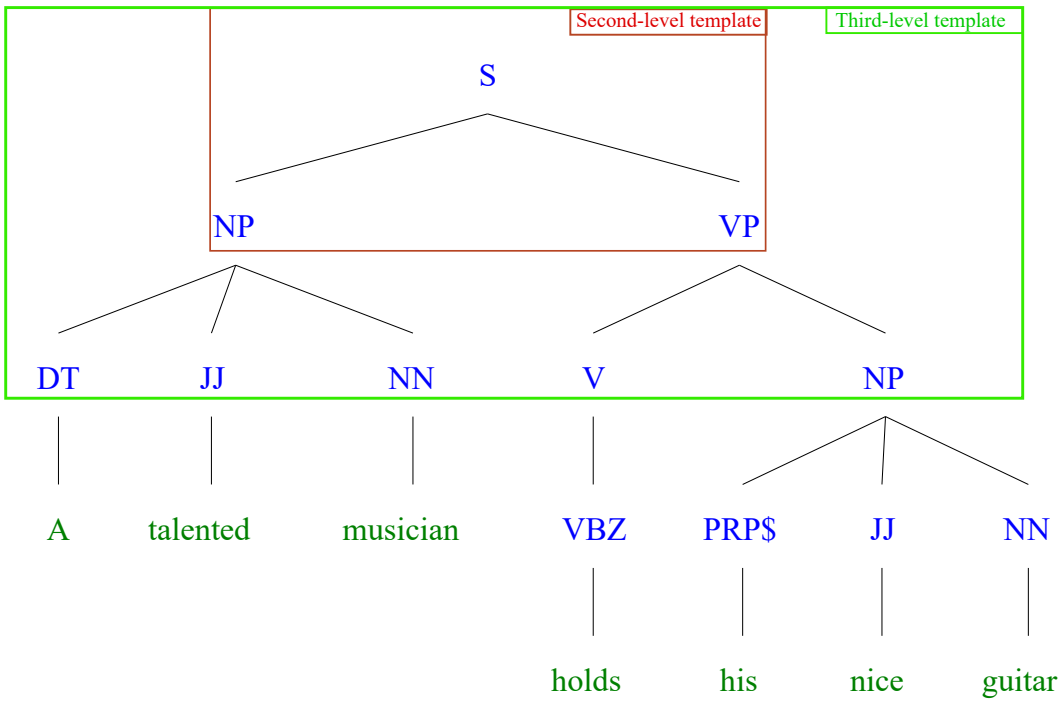


Figure 6.4: Example syntactic templates obtained by cutting a constituency tree at a certain level. The red frame delimits a syntactic tree cut at the 2^{nd} level (2^{nd} level template), while the green frame delimits a syntactic tree cut at the 3^{rd} level (3^{rd} level template).

structure, which is important to explain the intuition behind disentanglement with regard to syntactic roles in Chapter 8.

Part III

Semi-Supervised Learning with VAEs

Chapter 7

Challenging the Semi-Supervised VAE Framework for Text Classification

Obtaining labeled data to train NLP systems is a process that has often proven to be costly and time-consuming, and this is still largely the case [Böhmová et al., 2003, Martínez Alonso et al., 2016, Choudhary, 2018, Seddah et al., 2020]. Consequently, semi-supervised approaches are appealing to improve performance while alleviating dependence on annotations. To that end, Variational Autoencoders (VAEs; Kingma and Welling, 2014) have been adapted to semi-supervised learning [Kingma et al., 2014], and subsequently applied to several NLP tasks [Chen et al., 2018a, Corro and Titov, 2019, Gururangan et al., 2020]. As mentioned in 4.3, semi-supervised learning with VAEs is an instance of the case where supervised learning signal is used as an inductive bias to obtain a latent variable that has a clear meaning. This learning framework therefore yields an Autoencoder where the encoded representations are partly understandable, and where the decoded observations are decoded from this partly understandable representation, hence its importance for interpretability in general and for the present thesis in particular.

A notable difference between the generative model case from where VAEs originate, and the semi-supervised case is that only the decoder (generator) of the VAE is kept after training in the first case, while in the second, it is the encoder (classifier) that we keep. This difference, as well as the auto-regressive nature of text generators has not sufficiently been taken into account in the adaptation of VAEs to semi-supervised text classification. In this chapter, we show that some components can be ablated from the long used semi-supervised VAEs (SSVAEs) when only aiming for text classification. These ablations simplify SSVAEs and offer several practical advantages while preserving their performance and theoretical soundness.

The usage of unlabeled data through SSVAEs is often described as a *regularization* on representations [Chen et al., 2018a, Wolf-Sonkin et al., 2018, Yacoby et al., 2020]. More specifically, we explain in Section 7.1 that SSVAEs add to the supervised learning signal, a conditional generation learning signal that is used to train on unlabeled samples. From this observation, we study two changes to the standard SSVAE framework. The

first simplification we study (§ 7.1.2) is the removal of a term from the objective of SSVAEs: the Kullback-Leibler term. This encourages the flow of information into latent variables, frees the users from choosing priors for their latent variables, and is harmless to the theoretical soundness of the semi-supervised framework. The second simplification we study (§ 7.1.1) is made to account for the auto-regressive nature of text generators. In the general case, input samples in SSVAEs are described with two latent variables: a partially-observed latent variable, which is also used to infer the label for the supervised learning task, and an unobserved latent variable, which describes the rest of the variability in the data. However, auto-regressive text generators are powerful enough to converge without the need for latent variables. Therefore, removing the unobserved latent variable is the second change we study in SSVAEs. The above modifications can be found in some rare works throughout the literature, *e.g.* Corro and Titov [2019]. We, however, aim to provide justification for these changes beyond the empirical gains that they exhibit for some tasks.

Our experiments on four text classification datasets show no harm to the empirical classification performance of SSVAE in applying the simplifications above. Additionally, we show that removing the unobserved latent variable leads to a significant speed-up.

The contributions presented in this chapter are the following: we justify two simplifications to the standard SSVAE framework, we explain the practical advantage of applying them (§ 7.1), and we provide empirical results showing that they speed up the training process while causing no harm to the classification performance(§ 7.2).

7.1 Simplifying SSVAEs for Text Classification

As explained in § 3.7, semi-supervised learning with VAEs was introduced by Kingma et al. [2014]. Keeping the same terminology, we consider here a set of labeled examples $L = \{(x_1, y_1), ..., (x_{|L|}, y_{|L|})\}$, and a set of unlabeled examples $U = \{x'_1, ..., x'_{|U|}\}$. Also recall that *i)* besides the usual unobserved latent variable z, the semi-supervised VAE framework uses a partially-observed latent variable y; *ii)* the encoder $q_\phi(y|x)$ serves both as the inference module for the supervised task, and as an approximate posterior (and encoder) for the y variable in the VAE framework.

[Kingma et al., 2014] formulate the semi-supervised objective with L, U, x, y and z as follows:

$$\mathcal{J}^\alpha = \sum_{(x,y)\in L} \Big(\text{ELBo}((x,y);z) + \alpha \ \log q_\phi(y|x) \Big) \qquad + \sum_{x\in U} \text{ELBo}(x;(y,z)) \tag{7.1}$$

The first term in Equation 7.1 is the part used to train the network on the labeled data L, where ELBo considers x and y to be observed and z to be non-observed, and $q_\phi(y|x)$ is also trained with a Cross-Entropy objective weighted by a hyper-parameter α. The second term in $\mathcal{J}^\alpha$ is the term to be used with unlabeled samples from U, which is simply an ELBo that only considers x to be observed.

The simplifications we propose to the above SSVAE framework stem from an analysis of the alternative form under which ELBO can be written (Eq. 3.6 in Chapter 3). Although it is valid for any arguments of ELBo(.; .), we display it here for an observed variable x, and the couple of latent variables (y, z):

$$\text{ELBo}(x; (y, z)) = \log p_\theta(x) - \text{KL}[q_\phi(y, z|x)||p_\theta(y, z|x)] \tag{7.2}$$

For the case of SSVAEs, this form provides a clear reading of the additional effect of ELBo on the learning process: *i)* maximizing the log-likelihood of the generative model $p_\theta(x)$, *ii)* bringing the parameters of the inference model $q_\phi(y, z|x)$ closer to the posterior of the generative model $p_\theta(y, z|x)$. Since $p_\theta(y, z|x)$ is the distribution of the latent variables expected by the generative model p_θ for it to be able to generate x, we can conclude that ELBo trains *both* latent variables for conditional generation on the unsupervised dataset U. If we write $\text{ELBo}((x, y), z)$ under the same form:

$$\text{ELBo}((x, y); z) = \log p_\theta(x, y) - \text{KL}[q_\phi(z|x, y)||p_\theta(z|x, y)] \tag{7.3}$$

it can also be seen that only z is trained on conditional generation for the labeled examples in L.

7.1.1 Dropping the Unobserved Latent Variable

Building on observations from equations 7.2 and 7.3, we question the usefulness of training both latent variables for conditional generation when semi-supervised learning only aims for an improvement on the inference of the partially-observed latent variable y.

VAEs' generative capabilities have first been exhibited on image datasets [Kingma and Welling, 2014]. For such datasets, the reconstruction loss in ELBo is an L2 loss (resp. L1 loss). This stems from the fact that $p_\theta(x|z)$ is modeled by a Gaussian (resp. a Laplace) distribution. Given a single sample z, the blur modeled by such distributions cannot (in the general case) model an entire image dataset. The reader may refer to Zhao et al. [2017c] for a full discussion of this issue. As a consequence, it is necessary for such datasets to incorporate a latent variable z besides the partially-observed variable y to model the characteristics that y does not describe.

As explained in Chapter 3, most decoders used for language are auto-regressive, *i.e.* of the form $p_\theta(x|y, z) = \prod_i p_\theta(x_i|y, z, x_{<i})$. Such decoders are able to generate realistic samples when trained on a target text corpus, in fact so much that it causes them to ignore latent variables (*cf.* posterior collapse discussed in section 3.6). Given the above, $p_\theta(x) = \int_y p_\theta(x_0|y) \prod_i p_\theta(x_i|y, x_{<i})dy$, an autoregressive LM that incorporates only y as a latent variable is an expressive enough generative model to provide quality learning signal for y's training on conditional generation (Eq. 7.2). We therefore propose to keep only y and to drop z from the model avoiding its presence in the Kullback-Leibler divergence in Equation 7.2 and saving some parameters.

With this simplification, the training objective shown in Equation 7.1 becomes:

$$\sum_{(x,y)\in L} \Big(\log p_\theta(x|y) + \alpha \ \log q_\phi(y|x) \Big) \qquad + \sum_{x\in U} \text{ELBo}(x;y) \tag{7.4}$$

There is a caveat regarding this modification: Since using only y often makes integration over the latent variables possible[1], one may be tempted to optimize the exact log-likelihood instead of ELBo. This should not be done as it would remove the second term from Eq. 7.2, decoupling the learning processes of the generative model from that of the inference model, and therefore discarding the benefit provided by semi-supervised learning with VAEs. Nevertheless, keeping only y still often enables calculating exactly the reconstruction term which presents no harm to the learning process.

7.1.2 Dropping the Kullback-Leibler Term

As discussed in Section 3.6, the KL divergence in ELBo sometimes discourages the model from using latent variables and makes them useless in practice [Bowman et al., 2016, Zhao et al., 2017a, Chen et al., 2018c].

An interesting result from Zhao et al. [2017a] is that ELBo without KL divergence (*KL-free*) is still a theoretically sound objective for generative modeling with VAEs. The difference between the generative model resulting from a regular ELBo and a KL-free ELBo is the prior of the model. A KL-free ELBo results in a generative model that uses as a prior $q_\phi(z) = \int_z q_\phi(z|x)p_{data}(x)dx$. This prior is intractable which makes the resulting model impractical for generation, but causes no problem for semi-supervised VAEs. We therefore propose, as a second change to the standard SSVAE framework, the removal of the KL-divergence in ELBo.

Note that using the prior $q_\phi(z)$ mentioned above means that the network formulates its own prior instead of requiring the user to choose it. Priors for partially-observed latent variables are delicate to choose as it is highly preferred for them to *i)* yield a closed form or the KL-divergence in ELBo to stabilize training, *ii)* realistically model the *default* behavior of the latent variables. The latter requirement can be particularly tedious for non-trivial latent variable models (*e.g.* trees; Corro and Titov, 2019).

If we apply this removal of the KL-divergence to both ELBo's in the standard SSVAE objective $\mathcal{J}^\alpha$ in Equation 7.1, it becomes:

$$\begin{aligned} &\sum_{(x,y)\in L} \Big(\mathbb{E}_{z\sim q_\phi(z|x)} \left[\log p_\theta(x|y,z)\right] + \alpha \ \log q_\phi(y|x) \Big) \\ &+ \sum_{x\in U} \mathbb{E}_{(y,z)\sim q_\phi(y,z|x)} \left[\log p_\theta(x|y,z)\right] \end{aligned} \tag{7.5}$$

[1]For instance, integration becomes possible when the support of the discrete latent variable is small, as is the case for binary classification.

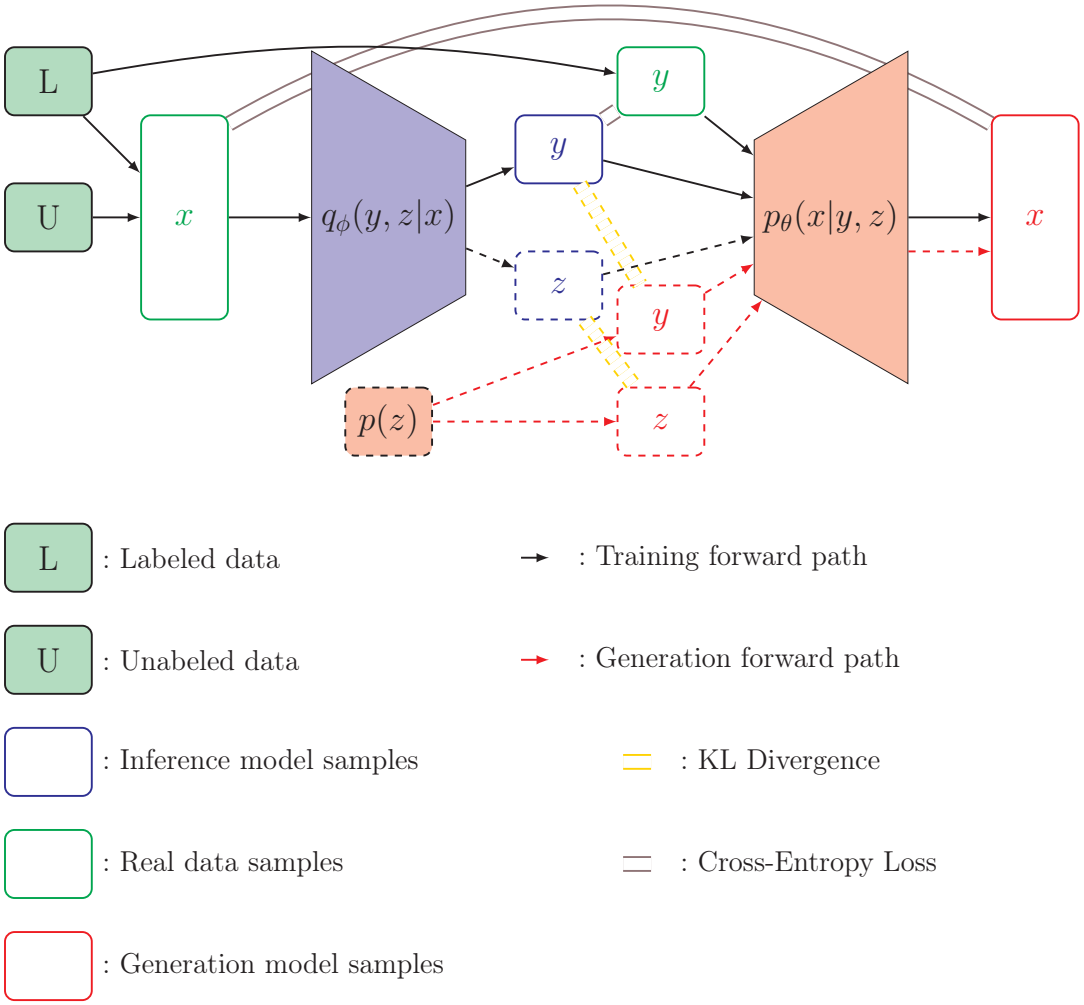


Figure 7.1: A diagram sketching the functioning scheme of a vanilla SSVAE. Dashed components are components we argue are unnecessary in our study.

7.1.3 Resulting Objective

Applying both of the previous simplifications to the semi-supervised objective in Eq. 3.24 leads to the following objective:

$$\sum_{(x,y)\in L} \Big(\log p_\theta(x|y) + \alpha \ \log q_\phi(y|x) \Big) + \sum_{x\in U} \mathbb{E}_{y\sim q_\phi(y|x)} \left[\log p_\theta(x|y) \right] \tag{7.6}$$

As can be seen, the first ELBo in Eq. 3.24 turns into a supervised conditional generation objective, while the second ELBo turns into a reconstruction term that relies only on y. Nevertheless, we stress that the second term is still an ELBo since removing the KL term still trains a generative model [Zhao et al., 2017a], and the whole objective is therefore still a VAE-based semi-supervised learning objective.

It should also be noted that, without z, the latent variables cannot provide the decoder with the full information about a sentence and, therefore, cannot reach a state where each sample is *reconstructed*. To avoid confusion, instead of *reconstructing* from y, the role of the reconstruction term is better read in our case as *raising the probability* of the sample at hand under the associated label y.

To better envision the impact of our modifications on the SSVAE framework, Figure 7.1 presents a holistic view of the ablations we presented in this section.

7.2 Experiments

In this section, we display comparisons between instances of standard SSVAEs and the same SSVAEs after applying the changes we propose.

7.2.1 Setup

Datasets We consider 4 datasets for our study: the IMDB [Maas et al., 2011] and Yelp review [Li et al., 2018] binary sentiment analysis datasets, and the AG News and DBPedia [Zhang et al., 2015] topic classification datasets. The datasets have been chosen to represent a range over different tasks (Sentiment Analysis and Topic Classification), different numbers of classes, and different sentence lengths. A summary of dataset statistics is in Table 7.1.

dataset	Labels	Av. Sample length	N° Classes
AG News	Topic	37.85±10.09	4
DBPedia	Topic	46.13±22.46	14
IMDB	Sentiment	233.79±173.72	2
Yelp	Sentiment	8.88±3.64	2

Table 7.1: Dataset properties.

As was done in Chen et al. [2020], we measure performance on the different datasets with equal numbers of samples. Accordingly, for each dataset, we randomly subsample 10K samples from the original training set as *unlabeled* data. We also use 4K labeled samples as training set and 1K as development set. We use the original test sets from each dataset. All the samples are tokenized using a simple whitespace tokenizer.

Network architecture The size of z is set to 32. For experiments without z, we simply drop all the components associated to it from the network.

The encoder consists of a pre-trained 300-dimensional fastText [Bojanowski et al., 2017] embedding layer, and 2 Bidirectional LSTM networks with 100 hidden states each, one for each of the latent variables y and z. The logits of y are then obtained by passing the

last state of its Bidirectional LSTM through a linear layer. Similarly the last state of the Bidirectional LSTM for z is passed through a linear layer to obtain its mean parameter, and a linear layer with a softplus activation to obtain its standard deviation parameter.

As for the decoding step, to allow backpropagation, z is sampled using the reparameterization trick [Kingma and Welling, 2014], and y is sampled using the Gumbel-Softmax trick [Jang et al., 2017]. Xu et al. [2017] have shown that latent variables are best exploited in SSVAEs when concatenated with the previous word at each generation step to obtain the next word. We design our decoder accordingly and use a 1-layered LSTM with size 200. The only hyper-parameter we tune on the development set is α, the coefficient weighting the supervised learning objective in Eq. 7.1, which is selected in the set $\{10^0, 10^{-1}, 10^{-2}, 10^{-3}\}$.

Training and validation data splits We sample 5 *labeled* data splits of size 1K. Each of these 5 splits will play, in turn, the role of validation set for one experiment, while the other 4 splits are used for training. Looping over these splits yields 5 runs for each experiment. The results we display are the average (and standard deviation) of the results for each of these runs. The validation score serves selecting hyper-parameters (in our case only α from Eq. 7.1). The final test scores are measured on the original test set of each dataset.

Probabilistic graphical model For models that use both z and y, we consider the latent variables to be conditionally independent in the inference model (*i.e.* $q_\phi(y, z|x) = q_\phi(y|x)q_\phi(z|x)$)) and independent in the generation model (*i.e* $p_\theta(y, z) = p(y)p(z)$).

Training procedure The network is optimized using ADAM [Kingma and Ba, 2015], with a learning rate of 4e-3 and a dropout rate of 0.5. If the accuracy on the validation set doesn't increase for 4 epochs, the learning rate is divided by 4. If it doesn't increase for 8 epochs, the training is stopped. For objectives that include a KL-divergence, we scale it with a coefficient that is null for 3K steps then linearly increased to 1 for the following 3K steps to avoid posterior collapse [Li et al., 2020a].

7.2.2 Results

Classification performance In Table 7.2, we compare the performance of a standard SSVAE, to a SSVAE where we remove the KL-divergence (SSVAE-{KL}) another where z is removed (SSVAE-{z}) and a third version where both the KL-divergence and z are removed (SSVAE-{KL, z}) for amounts of data varying from 1% to 100% of the 4K *labeled* samples in each dataset. We measure performance on all datasets using *accuracy*. As a baseline, we also include the results of an objective that does not use unlabeled data. The architecture we use for this objective is simply the LSTM encoder that we use to obtain y for the SSVAE objectives. This baseline is referred to as *Supervised*.

The aim of our experiment is to see whether we observe that there is a harm to the performance of SSVAEs when applying the proposed simplifications. In Table 7.2, we only

Dataset	Objective	1%	3%	10%	30%	100%
IMDB	Supervised	**54.62**(3.30)	56.47(1.02)	62.01(2.75)	69.65(2.02)	81.02(0.64)
	SSVAE	53.92(2.34)	56.03(4.20)	62.15(5.03)	75.39(0.49)	83.34(0.91)
	SSVAE-{KL}	52.70(1.72)	54.95(0.77)	62.37(4.45)	74.18(1.97)	83.87(0.47)
	SSVAE-{z}	54.15(2.46)	**56.86**(1.77)	62.15(2.87)	75.42(1.80)	81.90(5.17)
	SSVAE-{KL, z}	53.51(1.99)	56.58(2.22)	**63.24**(4.15)	**75.87**(1.30)	**84.79**(1.34)
AGNEWS	Supervised	**68.60**(4.88)	75.92(1.74)	81.96(0.83)	84.59(0.67)	86.98(0.74)
	SSVAE	65.79(5.02)	75.95(1.27)	82.47(0.43)	85.50(0.30)	87.89(0.54)
	SSVAE-{KL}	68.56(1.89)	76.25(2.21)	82.76(0.45)	85.73(0.80)	**87.95**(0.19)
	SSVAE-{z}	67.13(6.55)	**77.28**(1.81)	**83.48***(0.75)	**85.75**(0.74)	87.94(0.33)
	SSVAE-{KL, z}	66.96(3.42)	76.47(1.24)	82.58(0.97)	85.51(0.57)	87.85(0.29)
Yelp	Supervised	70.32(1.84)	76.32(2.07)	83.41(1.75)	87.85(0.58)	92.47(0.48)
	SSVAE	**71.34**(1.93)	76.96(1.64)	82.96(0.69)	89.35(0.39)	92.85(0.78)
	SSVAE-{KL}	69.85(2.86)	76.82(1.31)	82.90(2.23)	88.33(0.99)	92.90(0.54)
	SSVAE-{z}	68.74(2.95)	**78.26**(1.70)	84.11(1.25)	**90.27***(0.28)	93.60(0.74)
	SSVAE-{KL, z}	69.21(1.10)	77.30(2.57)	**85.02***(1.24)	89.74(1.31)	**93.77**(0.61)
DBPedia	Supervised	63.67(1.74)	81.49(2.25)	90.56(1.21)	94.63(0.32)	96.97(0.28)
	SSVAE	64.42(1.83)	83.16(1.49)	92.95(0.82)	96.26(0.25)	**97.75**(0.11)
	SSVAE-{KL}	**66.09**(3.05)	81.97(1.54)	**93.64**(0.76)	96.32(0.28)	97.58(0.13)
	SSVAE-{z}	62.56(5.60)	**83.40**(2.42)	93.37(1.00)	**96.39**(0.21)	97.40†(0.14)
	SSVAE-{KL, z}	62.15(1.68)	82.67(2.16)	93.40(1.10)	96.31(0.24)	97.58(0.19)

Table 7.2: Accuracies on IMDB, AGNEWS, Yelp and DBPedia with varying amount of **labeled** data. The values are averages over 5 runs with standard deviations between parentheses. The best score for each dataset and each amount of labeled data is given in bold. Each semi-supervised objective that scores above (resp. below) SSVAE with p-value$<$0.05 is marked with * (resp. †)

find 4 statistically significant differences between SSVAE and its variants: 3 in favor of one of our Simplified SSVAEs, and 1 in favor of the standard SSVAE.

Out-of-domain experiments The sentiment analysis tasks we use for these experiments take place in different domains (restaurant reviews for Yelp, and movie reviews for IMDB). Using models trained for each domain (with 100% of the data), we measure performance on the other domain to see whether the changes we study have an effect on out-of-domain generalization. In Table 7.3, we compare the out-of-domain performances of each of the objectives to that of the baseline that doesn't use unlabeled data (*Supervised*).

Dataset	Supervised	SSVAE	SSVAE-{KL}	SSVAE-{z}	SSVAE-{KL, z}
IMDB$\longrightarrow$Yelp	59.07(1.19)	61.78(6.03)	68.67^{+}(4.85)	**71.30**$^{+}$(7.67)	64.69^{+}(3.84)
Yelp$\longrightarrow$IMDB	66.17(2.62)	**69.54**(2.49)	66.67(3.26)	65.15(2.31)	66.13(3.82)

Table 7.3: Out-of-domain Accuracies between IMDB and Yelp for the different objectives. The best objective for each out-of-domain inference direction is given in bold. The scores displaying statistically significant improvement compared to the score of the supervised objective are marked with $^{+}$

The table shows no statistically significant gains from using unlabeled Yelp training

data for inference on IMDB. This is to be expected as reviews from Yelp are drastically shorter than those from IMDB (*cf.* Table 7.1). However, for out-of-domain inference in the opposite direction, all the semi-supervised objectives except the standard SSVAE show statistically significant gains. Removing the KL-divergence to accumulate more information in y, and removing z to have conditional generation exclusively rely on y seem to be effective to help generalization beyond the original domain of the task.

Speeding up the learning process By removing the KL-divergence and the components associated with z, an improvement on the speed of the learning process is to be expected. This improvement is highly dependent on the model and on the implementation at hand. As an example, we measure the average speed of an optimization iteration for each dataset, and each version of SSVAE. In Table 7.4, the speed of each objective is displayed proportionally to the speed of standard SSVAEs. The calculations associated with the KL-divergence do not seem to slow down the iterations. However, removing z and its associated components consistently cuts out a considerable proportion of the duration of optimization steps. This proportion ranges from 14% (DBPedia) to 26%(AGNEWS).

Dataset	SSVAE-{KL}	SSVAE-{z}	SSVAE-{KL, z}
AGNEWS	0.911(0.73)	**0.742**(0.65)	**0.742**(0.81)
DBPedia	1.03(0.56)	**0.861**(0.61)	0.867(0.56)
IMDB	1.018(0.25)	0.822(0.25)	**0.816**(0.25)
Yelp	0.986(1.52)	**0.819**(1.39)	**0.819**(1.52)

Table 7.4: Training durations for each objective relative to standard SSVAE, averaged over 200 iterations. Standard deviations are given between parentheses. Lowest duration for each dataset is given in bold.

7.3 Related Works

After the pioneering work of Kingma et al. [2014], SSVAEs were extended to tasks such as morphological inflections [Wolf-Sonkin et al., 2018], controllable speech synthesis [Habib et al., 2019], parsing [Corro and Titov, 2019], sequential labeling [Chen et al., 2018a] among many others. VAE internals have also been *tweaked* in various manners to improve the learning performance. For instance, Gururangan et al. [2020] introduce a low resource pretraining scheme to improve transfer with VAEs, while Zhang et al. [2019a] propose to use the deterministic ancestor of a latent variable to perform classification, and constrain it with an adversarial term to have it abide by the values of the random latent variable.

While this chapter is dedicated to the theoretical soundness and the practical advantages of two simplifications to the SSVAE framework for text classifications, it could be extendend to other tasks involving text generation as the unsupervised VAE objective.

For instance, the work of Corro and Titov [2019] shows that semi-supervised dependency parsing scores higher with both the changes we study.

7.4 Conclusion

Starting from the observation that SSVAEs can be viewed as the combination of a supervised learning signal with an unsupervised conditional generation learning signal, we show that this framework needs neither to include a KL-divergence nor an unobserved latent variable (z) when dealing with text classification (§ 7.1). We subsequently perform experimental comparisons between standard SSVAEs and simplified SSVAEs that indicate that our simplifications show no harm to performance both for in-domain classification and out-of-domain classification (§ 7.2).

Our changes provide a number of practical advantages. First, removing the KL-divergence frees practitioners from choosing priors for the variables they use, and allows information to flow freely into these variables. Second, removing the latent variable z from the computational graph speeds up computation and shrinks the size of the network. Despite their popularity, VAEs are often tedious to train for NLP tasks. In that regard, our simplifications should facilitate their usage in future works.

The inductive bias used to obtain an interpretable latent variable in this chapter is a supervised learning signal that trains y to abide by a certain label. Although semi-supervised learning minimizes the need for labels, it still requires annotated data. As discussed in 4.3, inductive bias comes in various forms, some of which require no supervised learning signal and only rely on structural patterns shared by the model and the data (*cf.* HoloGAN's example discussed in the same section). The following chapters explore these types of inductive bias where models are built to induce understandable concepts in their latent representations without labeled data.

Part IV

Unsupervised Disentanglement of Sentence Representations

Chapter 8

Towards Unsupervised Content Disentanglement in Sentence Representations via Syntactic Roles

This chapter presents our first contribution on unsupervised disentanglement of sentence representations, *i.e.* the process of separating information in neural representations of sentences along understandable axes without annotated data. As discussed in § 4.5, disentanglement in NLP has been mostly performed to separate the semantics (or content) in a sentence from characteristics such as style and structure in order to generate paraphrases [Chen et al., 2019a, Bao et al., 2019, John et al., 2020, Huang and Chang, 2021b, Huang et al., 2021b]. We show in this chapter that the information in the content itself can be separated with a VAE-based model, and that this separation can be accomplished without supervision or input syntactic information. The axes along which we demonstrate content separation are the lexical realizations of core syntactic roles. As explained in § 6.2, some syntactic roles, such as subjects and direct objects, are said to be *core* because they strongly relate to the predicate-argument structure of sentences, which makes them compelling axes to separate information in sentences. In order to study this separation, we present a framework including a model designed to disentangle information from the realizations of different core syntactic roles and an evaluation protocol aimed at measuring this disentanglement.

The model we introduce is an Attention-Driven VAE (ADVAE), which we train on raw text from the Stanford Natural Language Inference (SNLI) dataset [Schmidt et al., 2019], a dataset where sentences exhibit low syntactic variation. It draws its inspiration from attention-based Machine Translation models [Bahdanau et al., 2015, Luong et al., 2015]. Such models translate sentences between languages with different underlying structures and can be inspected to show a coherent alignment between spans from both languages, as shown in § 5.1. Our ADVAE uses Transformers [Vaswani et al., 2017], an attention-based architecture, to map sentences from a language to a fixed number of independent latent variables, then map these variables back to the same sentences. Although ADVAE could be used to study other attributes, we motivate it (§ 8.2.1) and therefore study it for the alignment of syntactic roles with latent variables.

Evaluating disentanglement with regard to spans is challenging. After training the model and only for evaluation, we use linguistic information (from an off-the-shelf dependency parser) to first extract syntactic roles from sentences, and then study their relation to latent variables. To study this relation on the ADVAE decoder, we repeatedly *i)* generate a sentence from a sampled latent vector *ii)* perturb this latent vector at a specific location *iii)* generate a sentence from this new vector and observe the difference. On the encoder side, we study the attention values to see whether each latent variable is focused on a particular syntactic role in input sentences. The latter procedure is only possible through the way our ADVAE uses attention to produce latent variables. To the best of our knowledge, we are the first to use this transparency mechanism to obtain quantitative results for a latent variable model.

We first justify our focus on syntactic roles in § 8.1, then we go over our contribution, which is threefold: *i)* We introduce the ADVAE, a model that is designed for *unsupervised* disentanglement of syntactic roles, and that enables analyzing the interaction between latent variables and observations through the values of attention (§ 8.2), *ii)* We design an experimental protocol for the challenging assessment of disentanglement over realizations of syntactic roles, based on perturbations on the decoder side and attention on the encoder side (§ 8.3), *iii)* Our empirical results show that our architecture disentangles syntactic roles better than standard sequence VAEs and Transformer VAEs and that it is capable of controlling realizations of syntactic roles separately during generation when trained on a dataset with regularly structured sentences (§ 8.4.1). To better characterize the limits of our model, we also provide results for hierarchical version of ADVAE(§ 8.4.2), and for our standard ADVAE when trained on Yelp, a dataset with more challenging syntactic variability (§ 8.4.3).

8.1 Syntactic Roles as Targets for Unsupervised Disentanglement

As presented in § 6.2, dependency parsing yields a tree, where edges are labeled with syntactic roles (or relations or functions) such as *nominal subject* (*nsubj*). The lexical realizations of these syntactic functions are textual spans and correspond to syntactic constituents. For instance, the lexical realization of the *direct object* (*dobj*) of the verb *holds* in our example sentence displayed in Figure 8.1 is the span *his nice guitar*, with *guitar* as head.

In our work, we focus[1] on verbal roots of sentences, their nominal subjects, and their direct or prepositional objects, *i.e. core* (as opposed to *oblique*) syntactic roles. These syntactic roles directly relate to the predicative structure (*cf.* § 6.2) which is a decomposition of the information in a sentence into a predicate and its arguments. With that in mind, we hypothesize in this work that core syntactic roles could constitute principal axes of variation for the information in sentences. Given the similarity between the way VAEs form latent variables and PCA (*cf.* § 4.2), and under the hypothesis that core syntactic

[1]Future research that takes interest in the finer-grained disentanglement of content may simply study a larger array of syntactic roles. Using our current system we display results including all syntactic roles in Appendix B.5.

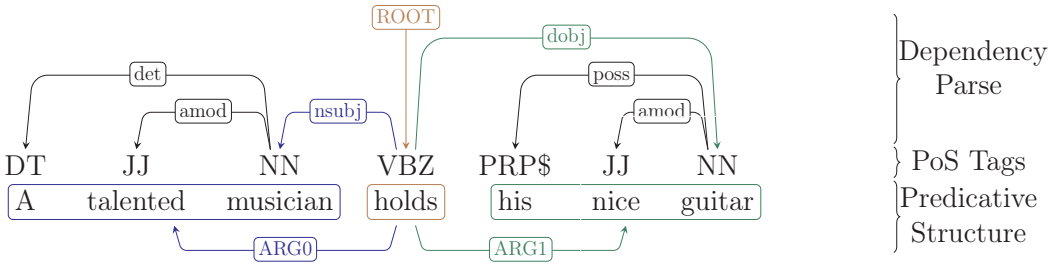


Figure 8.1: A sentence and its syntactic roles. The correspondence between syntactic roles and elements of the predicative structure is highlighted with colors. *ROOT* is the root node of the dependency tree, *nsubj* is a nominal subject, *dobj* is a direct object, *det* is a determiner, *poss* is a possessive determiner, and *amod* is an adjectival modifier. *ARG*0 and *ARG*1 are semantic *proto-roles* [Bonial et al., 2012] designating respectively the entity that performs the action (a.k.a the *agent*) and the entity affected by the action (a.k.a the *patient*).

roles are principal axes, we expect our VAE model to separate information pertaining to realizations of these syntactic functions without supervision, which is the goal of our work.

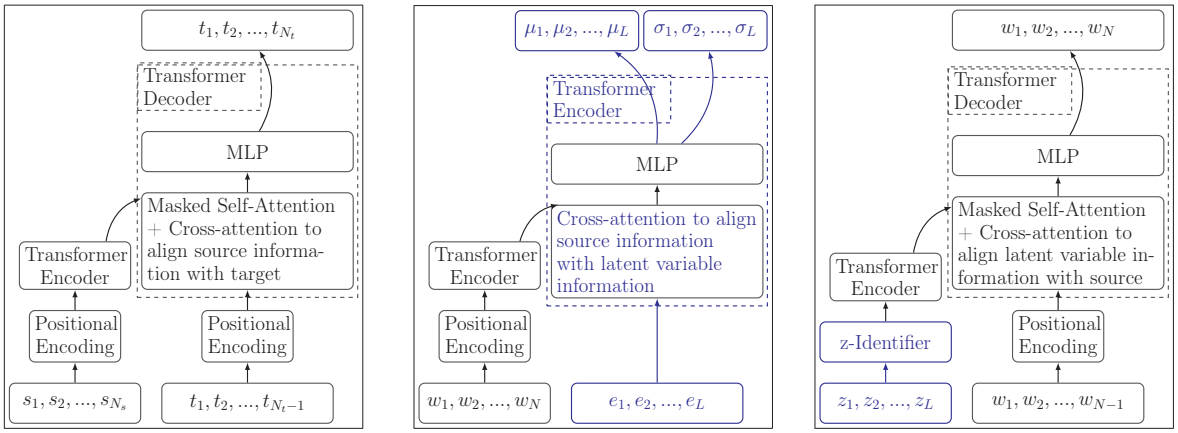
8.2 Model Description

The usual method to obtain sentence representations from Transformer models uses only a Transformer encoder either by taking an average of the token representations or by using the representation of a special token (*e.g* [CLS] in BERT [Devlin et al., 2019]). Recently, the usage of both Transformer encoders and decoders has also been explored in order to obtain representations whether by designing classical Autoencoders [Lewis et al., 2020, Siddhant et al., 2019, Raffel et al., 2020], or VAEs [Li et al., 2020b], where training involves Transformer encoders and decoders but representations are obtained with only the encoder. Our model, the ADVAE, differs from these models in that it uses both an encoder and a decoder to produce sentence representations, similar to the way a Machine Translation (MT) Transformers produces translations.

Producing representations with Cross-Attention has been introduced by Locatello et al. [2020c] as part of the Slot Attention modules in the context of unsupervised object discovery. However, in contrast to Locatello et al. [2020c], we simply use Cross-Attention as it is found in Vaswani et al. [2017], *i.e.* without normalizing attention weights over the query axis, or using GRUs [Cho et al., 2014b] to exchange information between the vectorial representations. As will be shown through our experiments, this is sufficient to disentangle syntactic roles. Concurrently to our work, Jaegle et al. [2022] also develop a model that uses Cross-Attention to encode inputs into latent vectors and to decode them with the intent to produce an architecture that is general enough to apply to all input/output format. Contrary to our model, their model *i)* does not use Self-Attention; *ii)* is not trained as a VAE, meaning it is not generative; *ii)* is not designed or evaluated for disentanglement.

We explain the observation that motivates our work in § 8.2.1, we then describe in

§ 8.2.2 the minimal changes we apply to MT Transformers, and finally, we present the objective we use in § 8.2.3. The parallel between our model and MT Transformers is illustrated in Figure 8.2.



(a) An MT Transformer (b) Our encoder (c) Our decoder

Figure 8.2: (a) is a minimalistic representation of the functioning scheme of an MT Transformer (full scheme can be seen in Figure 5.2).In blue, we highlight in (b) the difference between our encoder and a source-to-target MT model, and in (c) the difference between our decoder and a target-to-source MT model. The input at the bottom right for the Transformer Decoders in (a) and (c) is the series of previous words for auto-regressive generation. The input to our model is a series of words w, at the bottom left of (b), and its output is the reconstruction of these words in the same language, at the top right of (c).

8.2.1 The Intuition Behind our Model

Consider $s = (s_i)_{1 \leq i \leq N_s}$ and $t = (t_j)_{1 \leq j \leq N_t}$, two sequences of tokens forming respectively a sentence in the source language and a sentence in the target language. Given s, attention-based translation models are capable of yielding t while also providing information about the alignment between the groups of tokens (of different sizes) in both sentences [Bahdanau et al., 2015, Luong et al., 2015]). This evidence suggests that attention-based architectures are capable of factoring information from groups of words according to a source structure, and redistributing it according to a target structure.

The aim of our design is to use, as a target, a set of L *independent* latent variables that will act as fixed placeholders for the information in sentences. We stress that L is fixed and independent of the input sentence size N. Combining Transformers, an attention-based MT model, and the VAE framework for disentanglement, our ADVAE is intended to factor information from independent groups of words into separate latent variables. In the following sections, we refer to this set of independent latent variables as the latent *vector* $z = (z_l)_{1 \leq l \leq L}$ and to each z_l as a latent *variable*.

8.2.2 Model Architecture

Inference model: This is the inference model q_ϕ (encoder in Fig. 8.2.b) for our latent variables $z = (z_l)_{1 \leq l \leq L}$. It differs from an MT Transformer in two ways. First it uses as input a sentence w, and L learnable vectors $(e_l)_{1 \leq l \leq L}$ instead of the source and target

tokens s and t used in translation. The learnable vectors e will go through Cross-Attention without Self-Attention. We stress that these learnable vectors are input-independent. Second its output is not used to select a token from a vocabulary but rather passed to a linear layer (resp. a linear layer followed by a softplus non-linearity) to yield the mean parameters $(\mu_l)_{1\leq l\leq L}$ (resp. the standard deviation parameters $(\sigma_l)_{1\leq l\leq L}$) to parameterize the diagonal Gaussian distributions $(q_\phi^{(l)}(z_l|w))_{1\leq l\leq L}$.

Formally, let us define M^μ and M^σ to be linear layers that will respectively be used to obtain the latent variables' means and standard deviations. Given input token sequence w, the encoder $q_\phi(z|w) = \prod_l q_\phi(z_l|w)$ yields parameters μ_l and σ_l to be used by the diagonal Gaussian distribution of each of the latent variables z_l as follows[2]:

$$
\begin{aligned}
&\tilde{z} = \text{Dec}(e; \text{Enc}(w)) \\
&\forall\, l \quad \text{s.t. } 1 \leq l \leq L: \\
&\qquad \mu_l = M^\mu(\tilde{z}_l), \;\; \sigma_l = \text{SoftPlus}(M^\sigma(\tilde{z}_l)) \\
&\qquad z_l \sim \mathcal{N}(\mu_l; \sigma_l)
\end{aligned}
\tag{8.1}
$$

The distribution of the whole latent vector is simply the product of Gaussians $q_\phi(z_1, \ldots, z_L|s) = \prod_l^L q_\phi^{(l)}(z_l|w)$.

Generation model: Our generation model consists of an autoregressive decoder (Fig. 8.2.c) $p_\theta(w|z_1, \ldots, z_L) = \prod_i^N p_\theta(w_i|w_{<i}, z_1, \ldots, z_L)$ where $w_{<i}$ is the series of tokens preceding w_i, and a prior assuming independent standard Gaussian variables, *i.e.* $p(z_1, \ldots, z_L) = \prod_l^L p(z_l)$. Each latent variable z_l is concatenated with an associated learnable vector d_l (*z-Identifier* in Fig. 8.2.c) instead of going through positional encoding. From there on, the latent variables are used like source tokens in an MT Transformer.

For autoregressive decoding, Vaswani et al. [2017] define a version of Dec we call $\overline{\text{Dec}}$. This version uses Masked Self-Attention (Eq. 5.16 from Section 5.2) so that each word only queries information from the previously generated words. Even though $\overline{\text{Dec}}$ yields a sequence of length equal to that of the sentence w, in what follows, we consider its output to be only the last element of the sequence in order to express auto-regressive generation in a clear manner.

Given the above, Cross-Attention is used by the ADVAE decoder to dispatch information from the *source* latent variable samples to the *target* generated sequence. Accordingly, using a beginning-of-sentence token w_0, $p_\theta(w|z) = \prod_i p_\theta(w_i|w_{<i}, z)$ yields probabilities for the categorical distribution of the generated tokens w by decoding latent variables z con-

[2]The Dec and Enc modules used here were defined in § 5.2. To simplify equations, we omit word embedding look-up tables and positional encodings.

catenated with their embeddings d:

$$
\begin{aligned}
& y = \text{Cat}(d; z) \\
& \forall\, i \quad \text{s.t. } 1 \leq i \leq |w| : \\
& \qquad \tilde{w}_i = \overline{\text{Dec}}(w_0, \ldots, w_{i-1}; \text{Enc}(y)) \\
& \qquad w_i \sim \text{Categorical}(\text{softmax}(M^w(\tilde{w}_i)))
\end{aligned}
$$

8.2.3 Optimization Objective

We train our ADVAE using the β-VAE [Higgins et al., 2017] objective discussed in § 4.1:

$$\log p_\theta(w) \geq \mathbb{E}_{(z)\sim q_\phi(z|w)} \left[\log p_\theta(w|z)\right] - \beta\, \text{KL}[q_\phi(z|w)||p(z)] \tag{8.2}$$

In Eq. 8.2, w is a sample from our dataset, z is our latent vector and the distributions $p_\theta(w) = \int p_\theta(w|z)p(z)dz$ and $q_\phi(z|w)$ are respectively the generation model and the inference model. We use a standard Gaussian distribution as prior $p(z)$ and a diagonal Gaussian distribution as the approximate inference distribution $q_\phi(z|w)$. The weight β is used (as in Chen et al., 2018c, Xu et al., 2020b, Li et al., 2020c) to control disentanglement, but also to find a balance between the expressiveness of latent variables and the generation quality.

8.3 Evaluation Protocol

In order to quantify disentanglement, we first measure the interaction between latent variables and syntactic roles. To do so, we extract *core* syntactic roles from sentences according to the procedure we describe in § 8.3.1. Subsequently, for the ADVAE decoder, we repeatedly perturb latent variables and measure their influence on the realizations of the syntactic roles in generated sentences (§ 8.3.2). For the ADVAE encoder, we use attention to determine the syntactic role that participates most in producing the value of each latent variable (§ 8.3.3).

Given these metrics, we measure disentanglement taking inspiration from MIG in § 8.3.4. Recall from § 4.4 that MIG consists in measuring the difference between the first and second latent variables with the highest mutual information with regard to a target factor. It is intended to quantify the extent to which a target factor is concentrated in a single variable. This metric assumes knowledge of the underlying distribution of the target information in the dataset.However, there is no straightforward or agreed-upon way to set this distribution for text spans, and therefore to calculate MIG in our case. As a workaround, we use the influence metrics defined in § 8.3.2 and § 8.3.3 as a replacement for mutual information to quantify disentanglement.

8.3.1 Syntactic Role Extraction

We use the Spacy[3] dependency parser [Honnibal and Montani, 2017] trained on Ontonotes5 [Weischedel et al., 2013]. For each sentence the realization of *verb* is the root of the dependency tree if its POS tag is *VERB*. Realizations of *subj* (subject), *dobj* (direct object), and *pobj* (prepositional object) are *spans* corresponding to subtrees whose roots are labelled resp. *nsubj*, *dobj*, and *pobj* and which are direct children[4] of the verbal root.

A realization of a syntactic role in $R = \{verb, subj, dobj, pobj\}$ is empty if no node in the dependency tree satisfies its extraction condition.[5]

8.3.2 Latent Variable Influence on Decoder

Intuitively, we repeatedly compare the text generated from a sampled latent vector to the text generated using the same *vector* where only one latent *variable* is resampled. Thus we can isolate the effect of each latent *variable* on output text and gather statistics.

More precisely, we sample T^{dec} latent *vectors* $(z^{(j)})_{1\leq j\leq T^{dec}} = (z_l^{(j)})_{1\leq j\leq T^{dec}, 1\leq l\leq L}$. Then for each z^j, and for each l we create an altered version $\tilde{z}^{(jl)} = (\tilde{z}_{l'}^{(jl)})_{1\leq l'\leq L}$ where we resample only the l^{th} latent *variable* (*i.e.* $\forall l' \neq l,\ \tilde{z}_{l'}^{(jl)} = z_{l'}^{(j)}$).

Generating the corresponding sentences[6] with $p_\theta(w|z)$ yields a list of original sentences $(w^{(j)})_{1\leq j\leq T^{dec}}$, and a matrix of sentences displaying the effect of modifying each latent variable $(\tilde{w}^{(jl)})_{1\leq j\leq T^{dec}, 1\leq l\leq L}$. For each syntactic role $r \in R$, we denote the realization extracted from a sentence w with $\rho_r(w)$.

To measure the influence of a variable z_l on the realization of a syntactic role r, denoted Γ_{rl}^{dec}, we estimate the probability that a change in this latent variable incurs a change in the span corresponding to the syntactic role. We first discard, for the influence on a role r, sentence pairs $(w^{(j)}, \tilde{w}^{(jl)})$ where one the syntactic role is not realized in one of the sentences[7] , because the presence of a syntactic role is a property of its parent word, (*e.g.* the presence or absence of a *dobj* is controlled by the *transitivity* of the verb) hence not directly connected to the representation of the role r itself. As they are out of the scope of our chapter, we report measures of these structural changes (diathesis) in Appendix B.1, and leave their extensive study to future works. We denote the remaining number of samples T'^{dec}_{rl}.

In the following, we use operator $\mathbf{1}\{.\}$, which is equal to 1 when the boolean expression it contains is true and to 0 when it is false. This process yields a matrix Γ^{dec} of shape $(|R|, L)$ which summarizes interactions in the *decoder* between syntactic roles and latent

[3] https://spacy.io/models/en#en_core_web_sm

[4] *pobj* syntactic roles are taken to be the direct descendants of a preposition (*prep*) that is directly underneath the root.

[5] Examples of syntactic role extractions can be found in Appendix B.2.

[6] Throughout this chapter, we use greedy sampling, *i.e.* sampling the highest-probability word at each step as explained in § 2.3, for all generated sentences.

[7] In the original version of this work published in Felhi et al. [2021b], we discarded sentence pairs where a syntactic role appears or disappears. This causes sentence pairs that do not contain realizations of the syntactic role to count as sentences where the realization of the syntactic role did not change. This leads to some variations in the decoder-specific measures compared to the original publication. However, these variations do not alter the conclusions drawn from empirical results.

variables:

$$\Gamma^{dec}_{rl} = \sum_{j=1}^{T'^{dec}_{rl}} \frac{\mathbf{1}\{\rho_r(w^{(j)}) \neq \rho_r(\tilde{w}^{(jl)})\}}{T'^{dec}_{rl}} \tag{8.3}$$

8.3.3 Encoder Influence on Latent Variables

We compute this on a held out set of size T^{enc} of sentences $(w_i^{(j)})_{1\leq j\leq T^{enc}, 1\leq i\leq N_{w^{(j)}}}$. Each sentence $w^{(j)}$ of size $N_{w^{(j)}}$ generates an attention matrix $(a_{li}^{(j)})_{1\leq l\leq L, 1\leq i\leq N_{w^{(j)}}}$. Attention values are available in the Transformer encoder with Cross-Attention computing the inference model[8], and quantify the degree to which each latent variable embedding $e^{enc}_{z_i}$ draws information from each token w_j to form the value of z_i.

For the encoder, we consider the influence of a syntactic role on a latent variable to be the probability for the attention values of the latent variable to reach their maximum on the index of a token in that syntactic role's realization. The indices of tokens belonging to a syntactic role r in a sentence $w^{(j)}$ are denoted $\arg_r(w^{(j)})$. For each syntactic role r and sentence $w^{(j)}$, we discard inputs where this syntactic role cannot be found, and denote the remaining number of samples T'^{enc}_r. The resulting measure of influence of syntactic role r on variable z_l is denoted Γ^{enc}_{rl}. The whole process yields matrix Γ^{enc} of shape $(|R|, L)$ which summarizes interactions in the *encoder* between syntactic roles and latent variables:

$$\Gamma^{enc}_{rl} = \sum_{j=1}^{T'^{enc}_r} \frac{\mathbf{1}\{\underset{l}{\text{argmax}}(a_{li}^{(j)}) \in \arg_r(w^{(j)})\}}{T'^{enc}_r} \tag{8.4}$$

8.3.4 Disentanglement Metrics

For Γ^* (either Γ^{dec} or Γ^{enc}) each line corresponds to a syntactic role $r \in R$. The disentanglement metric for role r is the following:

$$\Delta\Gamma^*_r = \Gamma^*_{rm_1} - \Gamma^*_{rm_2} \tag{8.5}$$

$$s.t.\ m_1 = \underset{1\leq l\leq L}{\text{argmax}}\, \Gamma^*_{rl},\ \ m_2 = \underset{1\leq l\leq L, l\neq m1}{\text{argmax}}\, \Gamma^*_{rl} \tag{8.6}$$

We calculate total disentanglement scores for syntactic roles using Γ^{dec}, Γ^{enc} as follows:

$$\mathbb{D}_{dec} = \sum_{r\in R} \Delta\Gamma^{dec}_r\ ,\ \ \mathbb{D}_{enc} = \sum_{r\in R} \Delta\Gamma^{enc}_r \tag{8.7}$$

In summary, the more each syntactic role's information is concentrated in a single variable, the higher the values of $\mathbb{D}_{dec}$ and $\mathbb{D}_{enc}$. However, similar to MIG, these metrics do not say whether variables capturing our concepts of interest are *distinct*. Therefore,

[8]For simplicity, attention values are averaged over attention heads and transformer layers. This also allows drawing conclusions with regard to the tendency of the whole attention network, and not just particular specialized heads as was done in Clark et al. [2019]. For the sake of completeness, we display per-layer results in Appendix B.8.

we also report the number of distinct variables that capture the most each syntactic role (*i.e* the number of distinct values of m_1 in Eq. 8.5 when looping over r). This is referred to as $N_{\Gamma^{\text{enc}}}$ for the encoder and $N_{\Gamma^{\text{dec}}}$ for the decoder.

8.4 Experiments

8.4.1 Experimenting on Regularly Structured Sentences

Dataset Previous unsupervised disentanglement works [Higgins et al., 2017, Kim and Mnih, 2018, Li et al., 2020c] tend to use relatively homogeneous and low complexity data. The data has *low complexity* if it varies along clear factors which correspond to what the model aims to disentangle. Similarly, we use a dataset where samples exhibit low variance in terms of syntactic structure[9] while providing a high diversity of realizations for the syntactic roles composing the sentences, which is an adequate test-bed for unsupervised disentanglement of syntactic roles' realizations. This dataset is the plain text from the SNLI dataset [Bowman et al., 2015] extracted[10] by Schmidt et al. [2019]. The SNLI data is a collection of premises (on average 8.92 ± 2.66 tokens long) made for Natural Language Inference. We use 90K samples as a training set, 5K for development, and 5K as a test set.

Setup Our objective is to check whether the architecture of our ADVAE induces better syntactic role disentanglement. We compare it to standard Sequence VAEs [Bowman et al., 2016] and to a Transformer-based baseline that doesn't use Cross-Attention. Instead of Cross-Attention, this second baseline uses mean-pooling over the output of a Transformer encoder for encoding. For decoding, it uses the latent variable as a first token in a Transformer decoder, as is done for conditional generation with GPT-2 [Santhanam and Shaikh, 2019]. These comparisons are performed using the same β-VAE objectives, and the decoder disentanglement scores as metrics. Training specifics and hyper-parameter settings are detailed in Appendix B.3. For each of the two baselines, the latent variables we vary during the decoder's evaluation are the mono-dimensional components of its latent vector. It is easier to pack information about the realizations of multiple syntactic roles into D_z dimensions than into a single dimension. Consequently, the single dimensions we study for the baselines should be at an advantage to separate information into different variables.

Scoring disentanglement on the encoder side will not be possible for the baselines above as it requires attention values. To establish that our model effectively tracks syntactic roles, we compare it to a third baseline that locates each syntactic role through its median position across the dataset. This baseline is fairly strong on short sentences from a language where word order is rigid (*i.e* a configurational language) such as English. We refer to this Position Baseline as PB.

[9] Statistics regarding syntactic diversity for SNLI are provided in section 8.4.3 with a comparison to a more syntactically diverse dataset.

[10] github.com/schmiflo/crf-generation/blob/master/generated-text/train

The scores are given for different values of β (Eq. 8.2). Raising β lowers the expressiveness of latent variables, but yields better disentanglement [Higgins et al., 2017]. Following Xu et al. [2020b], we set β to low values to avoid posterior collapse. In our case, we observed that the models do not collapse for $\beta < 0.5$. Therefore, we display results for $\beta \in \{0.3, 0.4\}$. We stop at 0.3 as lower values for β result in poorer generation quality. For our model we report performance for instances with $L = 4$ (*ours-4*) and $L = 8$ (*ours-8*).

Results The global disentanglement metrics are reported in Table 8.1.[11]

Table 8.1: Disentanglement quantitative results for the encoder (enc) and the decoder (dec). N_Γ indicates the number of separated syntactic roles, and $\mathbb{D}$ measures concentration in a single variable. Values are averaged over 5 experiments. The standard deviation is between parentheses. PB refers to the Position Baseline.

Model	β	$\mathbb{D}_{enc} \uparrow$	$N_{\Gamma^{\text{enc}}} \uparrow$	$\mathbb{D}_{dec} \uparrow$	$N_{\Gamma^{\text{dec}}} \uparrow$
Sequence VAE	0.3	-	-	0.43(0.18)	1.70(0.48)
	0.4	-	-	0.91(0.32)	1.40(0.52)
Transformer VAE	0.3	-	-	0.08(0.04)	3.00(0.71)
	0.4	-	-	0.11(0.05)	3.80(0.45)
PB	-	0.98 (-)	3.00(-)	-	-
ours-4	0.3	1.48(0.15)	3.00(0.00)	0.78(0.10)	3.00(0.00)
	0.4	1.43(0.79)	3.00(0.00)	0.84(0.10)	3.00(0.00)
ours-8	0.3	1.34(0.18)	3.80(0.45)	0.62(0.17)	3.20(0.45)
	0.4	1.75(0.47)	2.80(0.45)	0.80(0.11)	3.00(0.00)

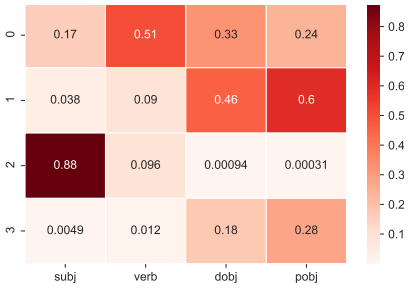


Figure 8.3: Encoder influence heatmap (Γ^{enc}).

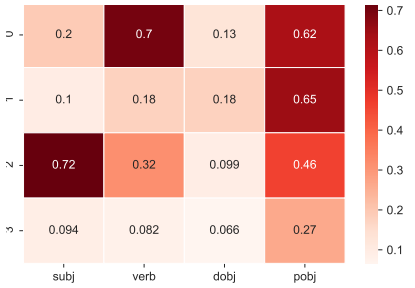


Figure 8.4: Decoder influence heatmap (Γ^{dec}).

On the decoder side, the Sequence VAE exhibits disentanglement scores in the range of those reported for our model for $\beta = 0.3$, and higher for $\beta = 0.4$. However, $N_{\Gamma^{\text{dec}}}$ shows that it controls the realizations of the 4 syntactic roles with less than 2 latent variables on average, meaning that it struggles to factor the realizations of different syntactic roles in different latent variables. The higher score shown for $\beta = 0.4$ is accompanied by an even

[11] Fine-grained scores are given in Appendix B.4.

Table 8.2: Resampling a specific latent variable for a sentence. The ID column is an identifier for the example.

ID	Original sentence	Resampled subject	Resampled verb	Resampled dobj/pobj
1	people are sitting on the beach	a young man is sitting on the beach	people are playing in the beach	people are sitting on a bench
2	a man and woman are sitting on a couch	a man is sitting on a park bench	a man and woman are running on a grassy field	the man and woman are on a beach
3	a man is playing with his dog	a boy is playing in the snow	a man is selling vegetables	a man is playing the game with his goal .

lower tendency to separate the information from different syntactic roles (*i.e.* lower $N_{\Gamma^{\text{dec}}}$). The Transformer VAE baseline assigns different latent variables to different syntactic roles (high $N_{\Gamma^{\text{dec}}}$), but suffers from very low specialization for these latent variables (low $\mathbb{D}_{dec}$). In contrast, our model is consistently able to separate 3 out of 4 syntactic roles, and a higher β raises its $\mathbb{D}_{dec}$. As *ours-8* has more latent variables, this encourages the model to further split the information in each syntactic role between more latent variables[12]. The fact that ADVAEs perform better than both Sequence VAEs and classical Transformer VAEs shows that its disentanglement capabilities are due to the usage of Cross-Attention to obtain latent variables, and not only to the usage of Transformers. On the encoding side, our models consistently score above the baseline, showing that our latent variables actively follow the syntactic roles.

In Figures 8.3 and 8.4, we display the influence matrices Γ^{enc} and Γ^{dec} for an instance of our ADVAE with $L = 4$ as heatmaps. The vertical axes correspond to the latent variables. As can be seen, our model successfully associates latent variables to verbs and subjects but chooses not to separate direct objects and prepositional objects into different latent variables. Upon further inspection of the same heatmaps for the Sequence VAE baseline, it appears that it most often uses a single latent variable for *verb* and *subj*, and another for *dobj* and *pobj*.

One can also notice in Figures 8.3 and 8.4, that the encoder matrix is sparser than the decoder matrix (which is consistent with the higher encoder disentanglement scores in Table 8.1). This is to be expected since the decoder $p_\theta(w|z)$ adapts the realizations of syntactic roles to each other after they are sampled separately from $p(z)$. The reason for this is that the language modeling objective requires some coherence between syntactic roles (conjugating verbs with subjects, changing objects that are semantically inadequate for a verb, etc). This *co-adaptation*, contradicts the independence of our latent variables. It is further discussed in the following paragraph.

[12]Results for a larger grid of L values are reported in Appendix B.9.

Table 8.3: Swapping the value of a specific latent variable between two sentences. The SSR (Swapped Syntactic Role) column indicates the syntactic role that has been swapped.

ID	Sentence 1	Sentence 2	SSR	Swapped Sentence 1	Swapped Sentence 2
1	a woman is talking on a train	people are sitting on the beach	subj	people are talking on a train	a woman is sitting on the beach
2	people are sitting on the beach	a woman is talking on a train	verb	people are talking on a beach	a woman is standing on a train
3	a woman is talking on a train	a man is playing with his dog	dobj/ pobj	a man is playing the guitar with a goal	a woman is performing a trick

Changing the realizations of syntactic roles Here, we display a few qualitative examples of how the realizations of syntactic roles can be separately changed using an instance of our ADVAE.

As a first example, we generate a sentence from a random latent vector, then resample for each syntactic role the corresponding disentangled latent variable to observe the change on the subsequently generated altered sentence. The results of this manipulation are in Table 8.2[13]. As can be seen, some examples exhibit changes that only affect the target syntactic role (example 1). However, the model often produces co-adaptations that go past the target syntactic role either for semantic soundness (example 2, resampled verb adapts the object), or simply for lack of generalization from the SNLI data used for training.

A second example we display is a swap of syntactic role realizations between sentences. A few examples are given in Table 8.3. Similar to Table 8.2, the model often yields the expected result. Co-adaptation is best seen here, as taking a syntactic role to a sentence with which it is incompatible results in unexpected changes (example 3).

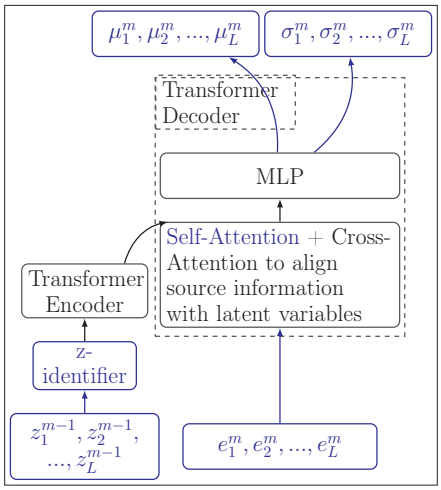
The co-adaptation seen here is caused by the independence between our latent variables which leaves it to the the decoder $p_\theta(w|z)$ to correct the incoherence between independently sampled syntactic role realizations[14]. Using structured latent variables to learn relations between syntactic roles seems to be the natural solution to this problem. Accordingly, the next section describes an investigation of a hierarchical version of the ADVAE.

8.4.2 A Hierarchical Version of our ADVAE

As we stated, our ADVAE aims to factor sentences into independent latent variables. However, given the dependency structure of sentences, realizations of syntactic roles are known to be interdependent to some degree in general. Therefore one may think that a structured latent variable model would be better suited to model the realizations of syntactic roles. In fact, such a model could absorb the language modeling co-adaptation

[13]More Examples are available in Appendix B.6

[14]We stress, here, that the co-adaptation we describe is different from entanglement. In fact, entanglement happens at the representation-level while co-adaptation can re-scramble, with the decoder, information that was separated in the representations.



(b)

Figure 8.5: The conditional inference module linking each of the hierarchy levels in our prior with the next level $p_\theta(z^m|z^{m-1})$. This module treats latent variables from previous layers as they are treated in our original decoder, and generates parameters for latent variables in subsequent hierarchy levels as it is done in our encoder.

between syntactic roles. For instance, instead of sampling an object and a verb from $p(z)$ that are inadequate, then co-adapting them through $p_\theta(w|z)$, a structured $p_\theta(z)$ could produce an *adequate* object for the verb. For this experiment, rather than using an independent prior $p(z)$, we use a structured prior $p_\theta(z) = p(z^0)\prod_{m=1}^{M} p_\theta(z^m|z^{m-1})$ where $p(z^0)$ is a standard Gaussian, and all subsequent $M-1$ hierarchy levels are parameterized by learned conditional diagonal Gaussians. The model used for each $p_\theta(z^m|z^{m-1})$ is shown in Figure 8.5.

We display the results for $M = 2$ and $M = 3$ in Table 8.4. For both models, we set L to 4.

Table 8.4: Disentanglement results for structured latent variable models on SNLI.

Depth	β	$\mathbb{D}_{enc}$	$N_{\Gamma^{\text{enc}}}$	$\mathbb{D}_{dec}$	$N_{\Gamma^{\text{dec}}}$
$M = 2$	0.3	0.79(0.36)	3.60(0.55)	0.69(0.22)	2.60(0.55)
	0.4	0.42(0.23)	2.80(0.45)	0.61(0.08)	2.40(0.55)
$M = 3$	0.3	0.90(0.25)	3.14(0.69)	0.60(0.18)	2.80(0.45)
	0.4	0.32(0.38)	2.75(0.50)	0.59(0.20)	3.20(0.84)

The results show lower mean disentanglement scores, and high standard deviations compared to the standard version of our ADVAE. By inspecting individual training instances of this hierarchical model, we found that some instances achieve disentanglement with close scores to those of the standard ADVAE, while others completely fail, which results in the high variances observed in Table 8.4. Unfortunately, hierarchical latent vari-

able models are notoriously difficult to train [Zhao et al., 2017b]. Our independent latent variable model is therefore preferable to the structured one due to these empirical results. More advanced hierarchical latent variable training techniques (such as Progressive Learning and Disentanglement [Li et al., 2020c]) may, however, provide better results. We plan to investigate this in our future works (*cf.* § 10.2).

8.4.3 Experimenting with the Yelp Dataset

As this is a first step in this research direction, we conducted this study on a dataset of relatively regular sentences. In this section, we aim to investigate the behavior of our ADVAE on the user-generated reviews from the Yelp dataset used in Li et al. [2018] using the same procedure we used for SNLI.

Contrasting SNLI to YELP: The length of sentences from this dataset (8.88 ± 3.64) is similar to the length of sentences from the SNLI dataset. However, contrary to SNLI, sentences presenting the regular Subject-Verb-Object (SVO) structure are much less common. To emphasize this difference, we label tokens in each dataset by the path that leads to them in the dependency tree and calculate the frequency (*i.e.* chances it appears in a sentence) of each one of such labels in SNLI and Yelp datasets. For example, in the sentence *"The man plays Football"*, *The* is labeled with *>nsubj>det*, meaning it is a determinant to a nominal subject beneath the root[15] of the sentence. In Figure 8.6, we display the top 30 labels in each dataset ranked by their frequency.

As can be seen in the figure, the frequency of core syntactic roles in SNLI is markedly higher than that of other remaining syntactic roles. Yelp, however, displays very different statistics where the core syntactic roles are much more rare, and where distributions of syntactic roles is much flatter, making it more difficult to learn syntactic regularities.

Results Similar to the experiments in the main body of the paper, we display the disentanglement scores in Table 8.5, and the influence metrics of one of the instances of our model as heatmaps in Figures 8.7 and 8.8.

Although the results show similar trends, they are weaker than what we obtained for SNLI. Given the difference between SNLI and Yelp (also discussed in Appendix B.2) there are two clear reasons for this decrease. The first is that Yelp is a dataset where it is harder to locate the syntactic roles. This is illustrated by the fact that the PB baseline obtains a much lower score. The second is that our syntactic role extraction heuristics are tailored for regular sentences with verbal roots, which subjects the evaluation metrics on Yelp to a considerable amount of noise. Nevertheless, the comparisons between a Sequence VAE, a Transformer VAE, an ADVAE, and PB retain the same conclusions, but with lower margins and some overlapping standard deviations.

Through manual inspection of examples, we observed that the occurrence of various syntactic phenomena (enumerations, sentences with nominal roots, presence of coordinating conjunctions, etc) was controlled by different latent variables. This calls for a model

[15] The root is ommitted in these labels for brevity.

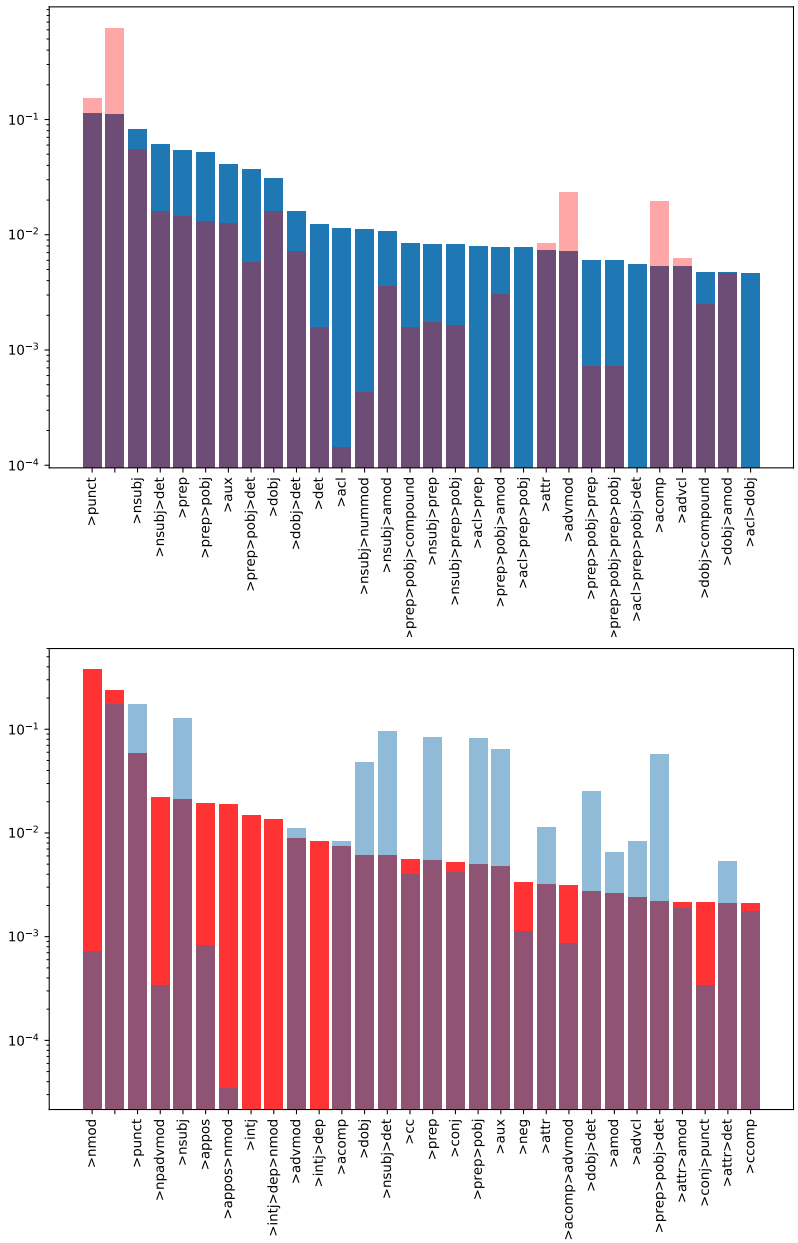


Figure 8.6: Top 30 dependency paths for each of SNLI (top, blue bars) and Yelp (bottom, red bars) datasets. In the top figure, the frequency of each dependency path among SNLI's top 30 is plotted in red with low opacity for Yelp to ease comparison. Analogously, low opacity blue bars are displayed in the bottom figure for SNLI's frequencies. The columns with an empty label corresponds to the root (empty) dependency path.

that provide ways to separate structural information, *i.e.* information pertaining to variations in syntax, from content-related information, *i.e.* information pertaining to variations in the lexical realization of the observed syntactic functions.

Table 8.5: Disentanglement results for the Yelp dataset

Model	β	$\mathbb{D}_{enc}$	$N_{\Gamma^{\text{enc}}}$	$\mathbb{D}_{dec}$	$N_{\Gamma^{\text{dec}}}$
Sequence VAE	0.3	-	-	0.49(0.04)	2.00(0.00)
	0.4	-	-	0.49(0.06)	1.8(0.84)
Transformer VAE	0.3	-	-	0.11(0.02)	2.80(0.45)
	0.4	-	-	0.11(0.05)	3.00(0.71)
PB	-	0.33(-)	2.00(-)	-	-
ours-4	0.3	0.48(0.07)	2.00(0.00)	0.23(0.09)	2.20(0.45)
	0.4	0.54(0.04)	3.00(0.00)	0.22(0.08)	2.20(0.45)
ours-8	0.3	0.44(0.04)	3.80(0.45)	0.17(0.09)	3.00(0.00)
	0.4	0.57(0.26)	3.40(0.55)	0.25(0.10)	2.80(0.84)

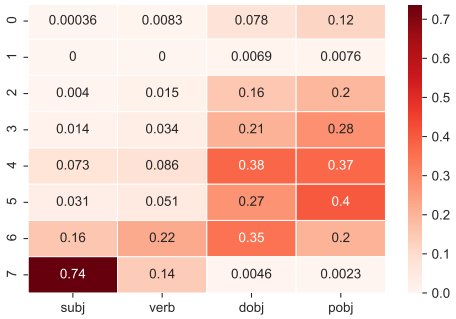


Figure 8.7: Encoder influence heatmap for Yelp(Γ^{enc}).

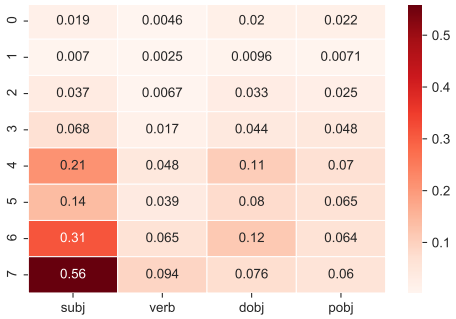


Figure 8.8: Decoder influence heatmap for Yelp(Γ^{dec}).

8.5 Related Works

Linguistic information in neural models Accounting for linguistic information so as to design neural networks with beneficial inductive bias has been a successful trend in NLP system design during recent years. For instance, successful attempts at capturing linguistic information with neural models helped improve grammar induction (RNNG; Dyer et al., 2016), constituency parsing and language modeling (ON-LSTM; Shen et al., 2019, ONLSTM-SYD; Du et al., 2020), as well as controllable generation (SIVAE; Zhang et al., 2019b). The evaluation protocol we present also relates our work to research that dives into the linguistic capabilities of neural NLP models (*cf.* 5.3.3). However, such studies most often rely on structural probes [Jawahar et al., 2019a, Liu et al., 2019, Hewitt and Manning, 2019b] to explain representations, probes which are not without issues, as shown by Pimentel et al. [2020b]. In that regard, the generative capabilities and the attention mechanism of our model offer an alternative to probing: analysis is performed directly on sentences generated by the model and on internal attention values.

Disentanglement in NLP As discussed in § 4.5, the main line of work in this area revolves around using multitask learning to separate concepts in neural representations (*e.g.* style vs content John et al., 2020; syntax vs semantics Chen et al., 2019a, Bao et al., 2019) and literature on *unsupervised* disentanglement in NLP remains scarce [Xu et al., 2020b, Behjati and Henderson, 2021]. The work of Behjati and Henderson [2021] is closest to ours as it uses Slot Attention [Locatello et al., 2020c], a Cross-Attention-based representation technique, to induce meaningful units from character sequences. Although developed with a different goal in mind, Perceiver IO [Jaegle et al., 2022], a work concurrent to ours, also shares similarities with our work given that it uses Cross-Attention to encode and decode latent vectors.

Our contribution differs from previous work in that *i)* syntactic parses are not used as learning signals but as a way to interpret our model, and *ii)* Cross-Attention enables our model to link a fixed number of latent variables to text spans.

8.6 Conclusion

The work presented in this chapter is the first part of our contribution on unsupervised disentanglement of sentence representations. Specifically, we study the hypothesis that lexical realizations of *core* syntactic roles can be disentangled *without supervision* (§ 8.1). Our framework includes: *i)* our model, the ADVAE (§ 8.2), which maps sentences to vectorial latent variables and allows for the use of attention to study the interaction between latent variables and spans; *ii)* an evaluation protocol to quantify disentanglement between latent variables and spans both in the encoder and in the decoder (§ 8.3).

Using our evaluation protocol we show in Section 8.4.1 that, when trained on a dataset of regularly structured sentences, ADVAE learns representations of sentences which exhibit a significant separation in the realizations of core syntactic functions without supervision. We also show that it separates syntactic roles to more latent variables than standard Sequence VAEs and with better concentration than standard Transformer VAEs.

This study constitutes a first step in a promising process towards *unsupervised* explainable modeling and fine-grained control over the lexical realizations of core syntactic roles in sentences. Although we focused on syntactic roles realizations, this architecture as well as the evaluation method are generic and could be applied to other tasks. For example, the architecture could be used at the document level (*e.g.* disentangling discourse relations), while the evaluation protocol could be applied to other spans such as constituents.

The limitations of ADVAE revealed in the last sections are twofold: *i)* a phenomenon we called *co-adaptation* where the independently sampled syntactic roles are subject to corrections by the decoder, which hurts our disentanglement metrics; *ii)* a degradation of disentanglement performance when dealing with a dataset that does not exhibit regular syntactic structures such as Yelp (§ 8.4.3). We have shown in section 8.4.2 that simply adding structure to our probabilistic graphical model so as to learn relations between our latent variables is not sufficient to deal with limitation *i)*. As for limitation *ii)*, it is

likely to originate from the fact that our model does not model syntactic variation. In the next chapter, we introduce a new latent variable that models syntactic information (again without supervision) and addresses this limitation.

Chapter 9

Exploiting Inductive Bias in Transformers for Unsupervised Disentanglement of Syntax and Semantics with VAEs

This chapter is the continuation of our work on unsupervised disentanglement of sentence representations. It focuses on a form of disentanglement that we discussed in Section 4.5 and that received a lot of interest from the NLP community: the separation between syntax and semantics in neural representations [Chen et al., 2019a,b, Bao et al., 2019, Zhang et al., 2019b, Huang and Chang, 2021a, Huang et al., 2021a]. The interest in this separation is mainly justified by the fact that purely semantic representations enable better paraphrase detection, and separation between syntax and semantics in an Seq2seq models enables paraphrase generation by changing syntax while keeping the same semantics. Previous works perform disentanglement using paraphrase pairs as information for semantics, and/or constituency parses as information for syntax. In general, the dependence of models on labeled data is known to often entail high cost [Böhmová et al., 2003, Martínez Alonso et al., 2016, Choudhary, 2018, Seddah et al., 2020], and to often require new labels to handle problems such as concept drift (*i.e.* changes in the relation between observation and label; Lu et al., 2019c) and domain adaptation (changes in the distribution of the observations; Farahani et al., 2021).

In light of the the difficulties incurred by the use of annotated data, we propose in this chapter an unsupervised model which directs syntax and semantics into different neural representations without semantic or syntactic information. In the Transformer architecture [Vaswani et al., 2017] presented in chapter 5, the attention mechanism is built upon a *query* from a set Q, which pools *values* V through *keys* K. For each query, values are selected according to their matching score computed by the similarity between their corresponding keys and the query. Building on an analogy between the (K, V) couple and syntactic roles with their lexical realizations (explicited in § 9.1.2) we present QKVAE[1], a Transformer-based VAE.

To build our model, we modify ADVAE, the model presented in the previous chapter.

[1] A contraction of the (Q, K, V) triplet with the VAE acronym.

Using Cross-Attention, QKVAE encodes sentences into two latent variables: z^{sem} to infer values for V, and z^{syn} to assign keys in K for values in V. These keys and values are then used in the attention mechanism of a Transformer Decoder to generate sentences. We show that z^{syn} tends to contain syntactic information, while z^{sem} tends to represent semantic information. Additionally, comparisons with a supervised model show that it needs a considerable amount of data to outperform our model on syntactic and semantic transfer metrics. Finally, we confirm the hypothesis formulated about syntactic information hindering syntactic role disentanglement with ADVAE's latent variables in the previous chapter, and show that QKVAE displays better syntactic role disentanglement metrics on the decoder side.

The contributions presented in this chapter can be summarized as follows:

- We describe QKVAE (§ 9.1), a model designed to disentangle syntactic information from semantic information by using separate latent variables for keys and values in Transformers attention.
- We run experiments on a dataset for English which empirically show that the two types of latent variables have strong preferences respectively for syntax and semantic (§ 9.2.2).
- We also show that our model is capable of transferring syntactic and semantic information between sentences by using their respective latent variables (§ 9.2.3). Moreover, we show that our model's ability to transfer syntax is competitive with supervised models when they use their full training set (more than 400k sentences), and that a supervised model needs a fairly large amount of labeled data (more than 50k samples) to outperform it on both semantic and syntactic transfer (§ 9.2.4).
- Looping back to syntactic role disentanglement (§ 9.3), we show that semantic latent variables in QKVAE, when freed of the syntactic information modeled by the syntactic variable, display better syntactic role disentanglement numbers on the decoder side than ADVAE's latent variables across 3 datasets which include datasets with non-regular syntactic structures[2].

9.1 QKVAE: Using Separate Latent Variables for Keys and Values

In this section, we describe the architecture of our model, the behavior it entails, and how we deal with the optimization challenges it poses.

9.1.1 QKVAE Architecture

The modification we bring to ADVAE is aimed at controlling how information is selected from the latent space with the value of a newly introduced latent variable. We call this latent variable z^{syn}, and refer to the latent variables already formulated in ADVAE as

[2]This contribution came after the publication corresponding to this chapter [Felhi et al., 2022], and was therefore not included in the publication.

$z^{sem} = \{z_1^{sem}, \dots, z_L^{sem}\}$. z^{syn} is obtained with the same process as each z_l^{sem} (Eq. 8.1), *i.e.* by adding an additional identifier embedding e_s, and matrices $M^{\mu s}$ and $M^{\sigma s}$ to obtain its mean and standard-deviation parameters.

For the QKVAE Decoder, we modify the Transformer Decoder Dec introduced in § 5.2 into QKVDec so as to use Multi-Head Attention with separate inputs for keys and values instead of Cross-Attention :

$$\text{QKVDec}(T; S_K; S_V) = \tilde{T}_{D^{QKV}}, \text{ s.t. :}$$
$$\tilde{T}_d = \begin{cases} T \text{ if } d = 0, \text{ else:} \\ \text{F}(\text{MHA}(\text{SA}(\tilde{T}_{d-1}), S_K, S_V) \end{cases}$$

where D^{QKV} is the number of layers. Recall, here, that theres a fixed number L of keys k and values v for all sentences regardless of their length. Similar to $\overline{\text{Dec}}$, we define $\overline{\text{QKVDec}}$ to be the auto-regressive version of QKVDec. The QKVAE decoder yields probabilities for the generated tokens by using this operator on values given by z^{sem} concatenated with embeddings d, and keys given by a linear transformation on z^{syn}:

$$\begin{aligned} & v = \text{Cat}(d; z^{sem}), \quad k = M^s(z^{syn}) \\ & \forall\, i \quad \text{s.t.} \quad 1 \leq i \leq |w| : \\ & \qquad \tilde{w}_i = \overline{\text{QKVDec}}(w_0, \dots, w_{i-1}; k; v) \\ & \qquad w_i \sim \text{Categorical}(\text{softmax}(M^w(\tilde{w}_i))) \end{aligned} \tag{9.1}$$

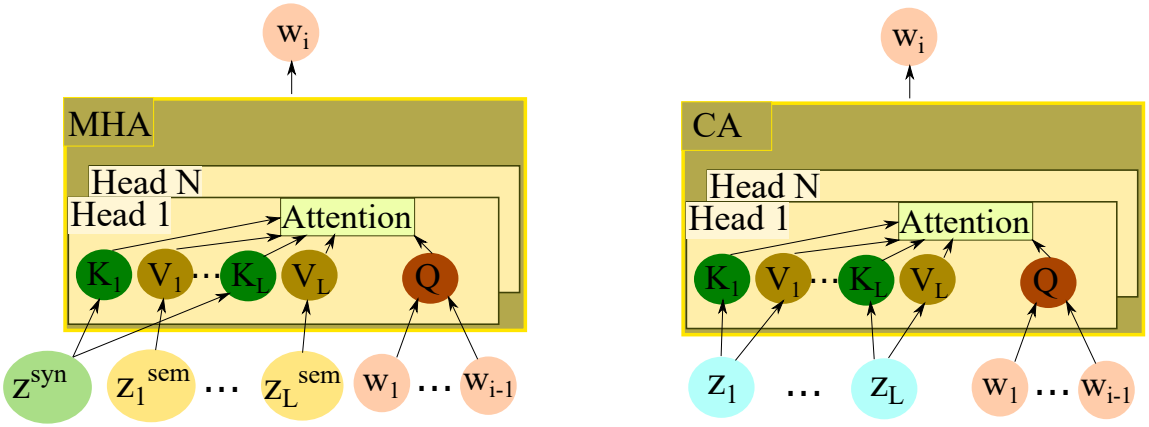


Figure 9.1: The usage of latent variables within ADVAE's decoder (right) and QKVAE's decoder (left). In contrast to ADVAE, all keys used during decoding in QKVAE come from a single latent variable that is separate from the latent variables used to obtain values.

where M^s is a linear layer.[3] While ADVAE already uses Cross-Attention to encode and decode latent variables, our model uses separate variables to obtain keys and values for Multi-Head Attention in its decoder. To better envision the difference between QKVAE

[3]The output of M^s is reshaped to obtain a matrix of keys.

and ADVAE, Figure 9.1 depicts the usage of latent variables in QKVAE and ADVAE decoders.

9.1.2 QKVAE Behavior

v	child	to wear	cloak	winter			
$k1$	nsubj	root	dobj	$\emptyset$	$\longrightarrow$	decoded (v, $k1$):	A child wears a cloak.
$k2$	agent	root	nsubjpass	pobj	$\longrightarrow$	decoded (v, $k2$):	A cloak is worn, in winter, by a child

Table 9.1: Example of interpretable values for the v and k in our model with $L = 4$. We display a sentence transiting from the active form to the passive form, to illustrate how different *keys* arranging the same *values* can lead to the same minimal semantic units being rearranged according to a different syntactic structure. We also stress that a different set of *keys* may omit or bring forth an element from the *values* vector (*e.g.* "winter" here above).

In the Multi-Head Attention of our decoder, z^{syn} controls keys, and z^{sem} controls values. In other words, the value of each z_l^{sem} is called to be passed to the target sequence according to its key k_l which is given by the variable z^{syn}. Therefore, for each query, z^{syn} decides which content vector z_l^{sem} participates most to the value of the generated token at each generation step. To better get a gist of the kind of behavior *intended* by this construction, we assume in Table 9.1 for explanatory purposes, that our decoder has one layer and one attention head, that the value of each k^l in key matrices k_1 and k_2 corresponds to syntactic roles, and that each v^l informs on the realization of the corresponding syntactic role. Table 9.1 displays the resulting sentence when each of $k1$ and $k2$ are coupled with v.

In the examples in Table 9.1, the generator uses a query at each generation step to pick a word in a manner that would comply with English syntax. Therefore, the key of each value should inform on its role in the target structure, which justifies syntactic roles as an adequate meaning for keys.

Although our model may stray from this possibility and formulate non-interpretable values and keys, keys will still inform on the *roles* of values in the target structure, and therefore influence the way values are injected into the target sequence. And given the fact that our model uses multiple layers and attention heads, and the continuous nature of keys in attention (as opposed to discrete syntactic role labels), our model performs a multi-step and continuous version of the behavior described in Table 9.1.

Injecting values into the structure of a sentence requires the decoder to model this structure. Previous works have shown that this is well within the capabilities of Transformers. Specifically, Hewitt and Manning [2019a] showed that Transformers embed syntactic trees in their inner representations, Clark et al. [2019] showed that numerous attention heads attend to specific syntactic roles, and we showed in the previous chapter that Transformer-based VAEs can capture the realizations of syntactic roles in latent variables obtained with Cross-Attention.

9.1.3 Balancing the Learning of z^{sem} and z^{syn}

Similar to ADVAE, we use a standard Normal distribution as a prior $p(z) = p(z^{sem})p(z^{syn})$ and train QKVAE with the β-VAE objective. To avoid posterior collapse, we follow the strategy from Li et al. [2020a] presented in § 3.6: *i)* We pretrain our model as an autoencoder by setting β to 0; *ii)* We linearly increase β to its final value (KL annealing; Bowman et al., 2016) and we threshold each dimension of the KL term with a factor λ (Free-Bits strategy; Kingma et al., 2016).

In preliminary experiments with our model, we observed that it tends to encode sentences using only z^{sem}. As we use conditionally independent posteriors[4] $q(z^{syn}|w)$ and $q(z^{sem}|w)$ for our latent variables, their KL terms can be written separately, and they can therefore be weighted separately with different values of β. Using a lower β for z^{syn} as was done by Chen et al. [2019b] [5] did not prove effective in making it informative for the model. Alternatively, linearly annealing β for z^{sem} before z^{syn} did solve the issue. This intervention on the learning process was inspired by the work of Li et al. [2020c] which shows that latent variables used at different parts of a generative model should be learned at different paces.

9.2 Experiments

9.2.1 Setup

Data To compare our model to its supervised counterparts, we train it with data from the English machine-generated paraphrase pairs dataset ParaNMT [Wieting and Gimpel, 2018]. More specifically, we use the 493K samples used by Chen et al. [2019b][6] to train their model VGVAE. Since our model is unsupervised, we only use the reference sentences (half the training set) to train our model. Using the development and test sets of ParaNMT, Chen et al. [2019b] also provide a curated set of triplets formed by a target sentence (*target*), a semantic source (*sem_src*),and a syntactic source (*syn_src*). The semantic source is a paraphrase of the target sentence, while the syntactic source is selected by finding a sentence that is syntactically close to the target (*i.e.* edit distance between the sequence of PoS Tags of both sentences is low[7]) and semantically different from the paraphrase (has low BLEU score with it). Contrary to paraphrases in the training set of ParaNMT, paraphrases from this set were manually curated. These triplets are divided into a development set of 500 samples and a test set of 800 samples. We display results on the test set in the main body of the paper. The results on the development set, which lead to the same conclusions, are reported in Appendix C.1.

Training details & hyper-parameters Encoders and Decoders in QKVAE are initialized with parameters from BART [Lewis et al., 2020]. After manual trial and error

[4] These posteriors are ADVAE encoders (Eq. 8.1).
[5] Although not explicitly mentioned in the paper, this is performed in their companion source code.
[6] https://drive.google.com/open?id=1HHDlUT_-WpedL6zNYpcN94cLwed_yyrP
[7] We follow Chen et al. [2019b] by using this evaluation data, although edit distance between PoS tags might not be a good proxy for syntactic similarity.

on the development set, we set the sizes of z^{syn} and z^{sem} to 768, and L to 4. Further Hyper-parameters are in Appendix C.2. We train 5 instances of our model and report the average scores throughout all experiments.

Baselines We compare our system to 4 previously published models, where 2 are supervised and 2 are unsupervised: *i) VGVAE [Chen et al., 2019b]:* a VAE-based paraphrase generation model with an LSTM architecture. This model is trained using paraphrase pairs and PoS Tags to separate syntax and semantics into two latent variables. This separation is used to separately specify semantics and syntax to the decoder in order to produce paraphrases; *ii) SynPG [Huang and Chang, 2021a]:* A paraphrase generation Seq2Seq model based on a Transformer architecture which also separately encodes syntax and semantics for the same purpose as VGVAE. This model is, however, trained using only source sentences with their syntactic parses, without paraphrases; *iii) Optimus [Li et al., 2020b]:* A large-scale VAE based on a fusion between BERT [Devlin et al., 2019] and GPT-2 [Radford et al., 2019] with competitive performance on various NLP benchmarks; *iv) ADVAE:* This model is QKVAE without its syntactic variable. The size of its latent variable is set to 1536 to equal the total size of latent variables in QKVAE.

Official open-source instances[8] of the 4 models above are available, which ensures accurate comparisons. The off-the-shelf instances of VGVAE and SynPG are trained on ParaNMT with GloVe[9] [Pennington et al., 2014] embeddings. We fine-tune a pre-trained Optimus on our training set following instructions from the authors. Similar to QKVAE, we initialize ADVAE with parameters from BART [Lewis et al., 2020] and train 5 instances of it on ParaNMT with $L = 4$.

9.2.2 Syntax and Semantics Separation in the Embedding Space

We first test whether z^{syn} and z^{sem} respectively specialize in syntax and semantics. A syntactic (resp. semantic) embedding should place syntactically (resp. semantically) similar sentences close to each other in the embedding space.

Using the (*target, sem_src, syn_src*) triplets, we calculate for each embedding the proportion of *target* sentences closer to *sem_src* than they are to *syn_src* in the embedding space. For simplicity, we refer to the syntactic and semantic embeddings of all models as z^{syn} and z^{sem}. For Gaussian latent variables, we use the mean parameter as a representation (respectively the mean direction parameter from the von Mises-Fisher distribution of the semantic variable of VGVAE). We use an Euclidean distance for Gaussian variables and a cosine distance for the others.

Since Optimus and ADVAE do not have separate embeddings for syntax and semantics *i)* We take the whole embedding for Optimus; *ii)*For ADVAE, we measure the above proportion on the development set for each latent variable z_l (Eq. 8.1). Then, we choose

[8]VGVAE: github.com/mingdachen/syntactic-template-generation/; SynPG: github.com/uclanlp/synpg; Optimus: github.com/ChunyuanLI/Optimus; ADVAE: github.com/ghazi-f/ADVAE

[9]Gains could be observed with better embeddings for supervised models, but we stick to the original implementations.

	$z^{sem} \uparrow$	$z^{syn} \downarrow$
Supervised Models		
VGVAE	99.9	14.8
SynPG	93.4	26.5
Unsupervised Models		
Optimus	91.8	-
ADVAE	39.5	40.0
QKVAE	89.2	26.4

Table 9.2: The proportion*100 of embeddings that place a target sentence closer to its semantic source than it is to its syntactic source in the embedding space. Arrows (↑/↓) indicate whether higher or lower scores are better.

the latent variable that places *target* sentences closest to their *sem_src* (resp. *syn_src*) as a semantic (resp. syntactic) variable. The results are presented in Table 9.2.

Table 9.2 clearly shows for QKVAE, SynPG, and VGVAE that the syntactic (resp. semantic) variables lean towards positioning sentences in the embedding space according to their syntax (resp. semantics). Surprisingly, the syntactic variable of our model specializes in syntax (*i.e.* has low score) as much as that of SynPG. The generalist latent variable of Optimus seems to position sentences in the latent space according to their semantics. Accordingly, we place its score in the z^{sem} column. Interestingly, the variables in ADVAE have very close scores and score well below 50, which shows that the entire ADVAE embedding leans more towards syntax. This means that, without the key/value distinction in the attention-based decoder, the variables specialize more in structure than in content.

9.2.3 Syntactic and Semantic Transfer

Similar to Chen et al. [2019b], we aim to produce sentences that take semantic content from *sem_src* sentences and syntax from *syn_src* sentences. For each of SynPG, VGVAE, and QKVAE we simply use the syntactic embedding of *syn_src*, and the semantic embedding of *sem_src* as inputs to the decoder to produce new sentences. Using the results of the specialization test in the previous experiment, we do the same for ADVAE by taking the 2 latent variables that lean most to semantics (resp. syntax) as semantic (resp. syntactic) variables. The output sentences are then scored in terms of syntactic and semantic similarity with *sem_src*, *syn_src* and *target*.

Control and reference baselines Beside model outputs, we also use our syntactic and semantic comparison metrics, explicited below, to compare *syn_src* and *sem_src* sentences to one another and to *target* sentences. Additionally, using Optimus, we embed *sem_src* and *syn_src*, take the dimension-wise average of both embeddings, and decode it. As VAEs are known to produce quality sentence interpolations [Bowman et al., 2016, Li et al., 2020b], the scores for sentences from Optimus help contrast a naïve fusion of features

	sem_src			*syn_src*			*target*		
	STED↑	*TMA2*↓	*TMA3*↓	*STED*↓	*TMA2*↑	*TMA3*↑	*STED*↓	*TMA2*↑	*TMA3*↑
Control and Reference baselines									
sem_src	0.0	100	100	13.0	40.3	4.8	12.0	39.6	7.0
syn_src	13.0	40.3	4.8	0.0	100	100	5.9	84.3	45.8
Optimus	11.6	50.0	15.9	9.2	61.6	23.6	10.2	58.9	21.8
Supervised Models									
VGVAE	13.1	39.9	5.4	3.3	86.4	64.1	6.7	80.4	44.6
SynPG	11.7	41.9	18.0	13.5	74.1	10.5	13.1	69.1	13.3
Unsupervised Models									
ADVAE	11.9	47.3	14.0	10.3	54.3†	19.2†	11.1	52.3	17.0
QKVAE	12.7	40.2	7.8	7.2	68.2	39.5	8.9	63.9	28.1

Table 9.3: Syntactic transfer results. *STED* is the Syntactic Tree Edit Distance, and *TMA2/3* is the exact matching between constituency trees truncated at the $2^{nd}/3^{rd}$ level. The comparison scores between sentences and *syn_src* that are not significantly different from the same scores produced with regard to *sem_src* are marked with †. We consider differences to be significant if their associated t-test yields a p-value$<$0.01.

in the embedding space with a composition of well identified disentangled features.

Transfer metrics We measure the syntactic and semantic transfer from source sentences to output sentences. *i) Semantics:* For semantics, previous works [Chen et al., 2019b, Huang and Chang, 2021a] rely on lexical overlap measures such as BLEU [Papineni et al., 2001], ROUGE [Lin, 2004], and Meteor [Denkowski and Lavie, 2014]. As will be shown in our results, the lexical overlap signal does not capture semantic transfer between sentences when this transfer is too weak to produce paraphrases. Therefore, we use Meteor (M) in conjunction with ParaBART [Huang et al., 2021a] a model where BART [Lewis et al., 2020] is fine-tuned using syntactic information to produce neural representations that represent semantics maximally and syntax minimally. We measure the cosine similarity between sentences according to ParaBART embeddings (PB). *ii) Syntax:* We use the script of Chen et al. [2019b] to produce a syntactic tree edit distance (STED) between the constituency trees of sentences, as was done to assess VGVAE. Additionally, following the evaluation procedure designed by Huang and Chang [2021a] for SynPG, we measure the Template Matching Accuracy between sentences, where the template is the constituency tree truncated at the second level (TMA2). TMA2 is the percentage of sentence pairs where such templates match exactly. We extend this measure by also providing it at the third level (TMA3)[10]. Transfer results are presented in Tables 9.3 and 9.4.

Sanity checks with metrics and baselines We notice in Table 9.4 that using Meteor as a semantic similarity measure results in various inconsistencies. For instance, paraphrases *target* have a higher Meteor score with the syntactic sources than with interpolations from *Optimus*. It can also be seen that the Meteor score between outputs from VGVAE and both syntactic and semantic sources are rather close [11]. In contrast,

[10]For example cuts of constituency trees at the second or third level, refer to Figure 6.4 in Chapter 6.

[11]This was not observed by Chen et al. [2019b], as they only compared outputs from VGVAE to the target paraphrases.

	sem_src		*syn_src*		*target*	
	M↑	*PB*↑	*M*↓	*PB*↓	*M*↑	*PB*↑
Control and Reference baselines						
sem_src	100	1.0	6.9	0.14	28.8	0.84
syn_src	6.9	0.14	100	1.0	12.1	0.16
Optimus	12.4	0.34	15.9	0.39	10.8	0.32
Supervised Models						
VGVAE	17.6	0.58	15.3	0.18	24.9	0.58
SynPG	45.9	0.87	8.0	0.13	25.2	0.75
Unsupervised Models						
ADVAE	8.0	0.19	8.3†	0.17	7.4	0.19
QKVAE	12.8	0.35	11.0	0.19	12.6	0.34

Table 9.4: Semantic transfer results. *M* is the Meteor score, and *PB* is the ParaBart cosine similarity. The comparison scores between sentences and *syn_src* that are not significantly different from the same scores produced with regard to *sem_src* are marked with †.

ParaBART score behaves as expected across comparisons in Table 9.4. Consequently, we retain ParaBART score as a semantic similarity measure. In the following, we use the scores between *sem_src*, *syn_src*, and *target* (first two rows in Tables 9.3 and 9.4) as reference scores for unrelated sentences, paraphrase pairs, and syntactically similar sentences.

Comparing the supervised baselines VGVAE and SynPG greatly differ in scores. It can be seen that SynPG copies a lot of lexical items from its semantic input (high Meteor score) which allows for higher semantic similarity scores. However, Table 9.3 shows that SynPG transfers syntax from *syn_src* at a high level (high TMA2, but low TMA3). In contrast, VGVAE transfers syntax and semantics in a balanced way and achieves the best syntax transfer scores overall (lowest STED with *syn_src* and *target*).

Analysing the scores of QKVAE The semantic similarity scores *PB* of QKVAE outputs with *target* and *sem_src* are close to those of Optimus outputs. Although these scores are low compared to supervised models, they are notably higher than semantic similarity scores between unrelated sentences (*e.g. syn_src* and *sem_src*). However, in contrast to Optimus, QKVAE outputs display low PB scores with *syn_src*, which show that they draw very little semantic information from the syntactic sources. Concerning syntactic transfer in Table 9.3, QKVAE outputs share syntactic information with *syn_src* on all levels (low STED, and high TMA2 and TMA3). Our model is even competitive with SynPG on TMA2, and better on TMA3 and STED. As expected, the scores comparing QKVAE outputs to *sem_src* show that they share very little syntactic information. On the other hand, ADVAE shows poor transfer performance on syntax and semantics, with only slight differences between scores w.r.t *syn_src* and scores w.r.t *sem_src*.

9.2.4 Comparison with a Supervised Model with Less Data

Since VGVAE displays balanced syntactic and semantic transfer capabilities, we use it for this experiment where we train it on subsets of sizes in $\{10K, 25K, 50K, 100K\}$ from its original training data. Our goal is to find out how much labeled data is needed for VGVAE to outperform our unsupervised model on both transfer metrics.

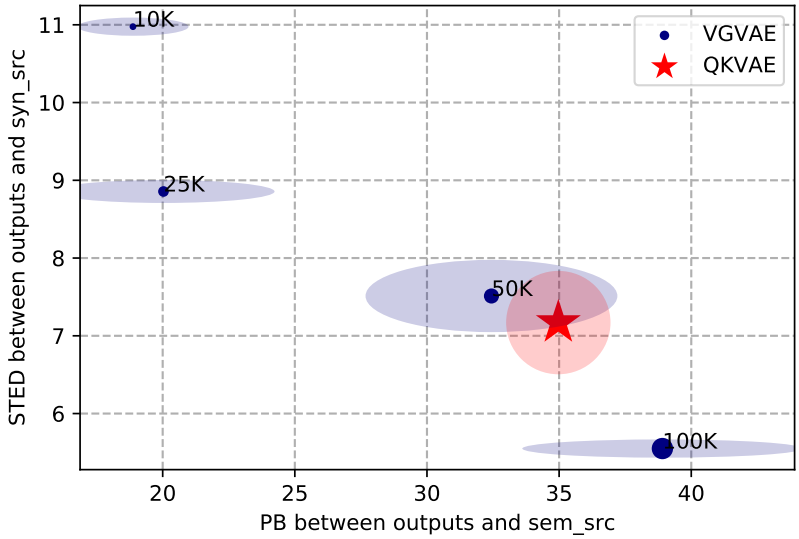


Figure 9.2: Plotting STED w.r.t *syn_ref* and the PB cosine similarity w.r.t *sem_ref* for VGVAE with different amounts of labeled data and for QKVAE. Points are scaled proportionally to the amount of training data. The vertical and horizontal diameters of each ellipse are equal to the standard deviation of the associated data points and axes.

In Figure 9.2, we plot for QKVAE and instances of VGVAE the *STED* of their outputs w.r.t *syn_src* and the *PB* of these outputs w.r.t *sem_src*. All values are averages over 5 runs, with standard deviations plotted as ellipses. Figure 9.2 shows that to outperform QKVAE on syntactic and semantic transfer, VGVAE needs more than 50K labeled samples.

9.3 A Look-Back to Syntactic Role Disentanglement

In the previous chapter, we saw that ADVAE performed poorly when it came to syntactic role disentanglement outside of the case where the dataset it was trained on was regularly structured, *e.g.* on the Yelp dataset instead of the SNLI dataset. We speculated that latent variables in ADVAE also had to model syntax, and that accordingly, syntactic variation coming from re-sampling the latent variables made it very difficult to observe change focused on fixed syntactic roles.

In this chapter, we partly confirmed that speculation through the previous investigations. In fact, Table 9.2 showed that the vectorial latent variables in ADVAE leaned slightly more to syntax than they did to semantics. This shows that syntactic information does indeed hinder encoding semantic content in ADVAE. In the same table, we can see that those latent variables, when accompanied with a new variable which deals with

syntactic information as is done in QKVAE, display a much higher specialization in semantics. Given that the information encoded by QKVAE's z^c is much more semantic than the information encoded by ADVAE's z, chances are that QKVAE can better disentangle information about realizations of syntactic roles than ADVAE.

Setup: We run the same quantitative syntactic role disentanglement experiment as the one displayed in the previous chapter, while setting QKVAE to use the same hyperparameters and architecture as ADVAE from the previous chapter. We use $L = 4$ since results from the last chapter have shown that splitting information into more latent variables than the target syntactic roles generally leads to worse disentanglement. We train and measure performance on SNLI, Yelp and and a 1.5 Million sentences extract from Wikipedia.

Results: The results are displayed in Table 9.5.

data	model	beta	$\mathbb{D}_{enc}$	$N_{\Gamma^{\text{enc}}}$	$\mathbb{D}_{dec}$	$N_{\Gamma^{\text{dec}}}$
SNLI	ADVAE	0.3	1.48(0.15)	3.00(0.00)	0.78(0.10)	3.00(0.00)
		0.4	1.43(0.79)	3.00(0.00)	0.84(0.10)	3.00(0.00)
	QKVAE	0.3	1.06(0.09)	3.80(0.45)	1.09(0.17)	3.00(0.00)
		0.4	1.06(0.18)	3.00(0.00)	1.31(0.14)	3.00(0.00)
Wiki	ADVAE	0.3	0.41(0.30)	2.00(0.00)	1.72(0.61)	1.00(0.00)
		0.4	0.25(0.29)	2.67(0.58)	0.44(0.50)	2.00(1.00)
	QKVAE	0.3	0.32(0.06)	2.80(0.45)	0.69(0.18)	2.20(0.45)
		0.4	0.41(0.03)	3.00(0.00)	0.62(0.07)	2.33(0.58)
Yelp	ADVAE	0.3	0.48(0.07)	2.00(0.00)	0.23(0.09)	2.20(0.45)
		0.4	0.54(0.04)	3.00(0.00)	0.22(0.08)	2.20(0.45)
	QKVAE	0.3	0.45(0.07)	2.80(0.45)	1.05(0.11)	2.40(0.55)
		0.4	0.46(0.03)	2.60(0.55)	0.87(0.15)	2.40(0.55)

Table 9.5: Syntactic role disentanglement results for QKVAE vs ADVAE. For QKVAE, the indicated β value is used for both latent variables z^s and z^c.

On the encoder side, the table displays comparable or better performance for ADVAE compared to QKVAE across the 3 datasets. This is to be expected as the semantic latent variables of QKVAE will look at different syntactic roles for sentences with different syntax.

On the decoder side, the disentanglement measure for QKVAE involves only resampling one z^c_i while keeping z^s and any other z^c_j s.t. $j \neq i$ fixed, which should enable it to vary separately syntactic role realizations while better maintaining a fixed syntax for sentences. The numbers in Table 9.5 confirm this hypothesis by displaying consistently

higher[12] disentanglement concentration scores $\mathbb{D}_{dec}$ for QKVAE than it does for ADVAE across the 3 datasets with a considerable margin, especially for Yelp. The number of disentangled latent variables $N_{\Gamma^{\text{dec}}}$ is also either equal or slightly higher for QKVAE than it is for ADVAE.

9.4 Related Work

We broadly divide recent works on explainability in NLP into two research directions, where the first seeks *post hoc* explanations for black-box models (*cf.* § 5.3.3) and the second seeks to build models that are explainable by design. This led to models with explicit linguistically informed mechanisms such as the induction of grammars (RNNG; Dyer et al., 2016, URNNG; Kim et al., 2019) or constituency trees (ON-LSTM; Shen et al., 2019, ONLSTM-SYD; Du et al., 2020).

As a work on disentangled representation learning, this work belonds to this second research direction. As explained in § 4.5, disentanglement in NLP was performed on various characteristics in text such as style [John et al., 2020, Cheng et al., 2020], sentiment and topic [Xu et al., 2020b], or word morphology [Behjati and Henderson, 2021], with a particular focus on the separation between syntax and semantics, whether merely to obtain an interpretable specialization in the embedding space [Chen et al., 2019a, Bao et al., 2019, Ravfogel et al., 2020, Huang et al., 2021a], or for controllable generation [Chen et al., 2019b, Zhang et al., 2019b, Huang and Chang, 2021a, Hosking and Lapata, 2021, Li et al., 2021a, Hosking et al., 2022]. However, all these works rely on syntactic information (constituency parses and PoS tags) or semantic information (paraphrase pairs). To the best of our knowledge, our work is the first to present a method that directs syntactic and semantic information into assigned embeddings in the challenging unsupervised setup.

From a broader machine learning perspective, using knowledge of the underlying phenomena in our data, we design our model QKVAE with an inductive bias that induces understandable behavior in an unsupervised fashion. Among the existing line of applications of this principle [Rezende et al., 2016, Hudson and Manning, 2018, Locatello et al., 2020c, Tjandra et al., 2021], ADVAE [Felhi et al., 2021b], the model presented in Chapter 8 which constitutes the basis for QKVAE, is designed to separate information from the realizations of different syntactic roles without supervision on a dataset of regularly structured sentences.

9.5 Qualitative Results and Discussion

In Table 9.6, we display example outputs of SynPG, VGVAE, and QKVAE along with their syntactic sources, semantic sources, and targets. We generally observed that the outputs of QKVAE range from paraphrases (line 6) to broadly related sentences (line 3).

As was shown by our quantitative results, outputs from VAE-based models (VGVAE and QKVAE) share relatively few lexical items with the semantic input. This can be seen

[12]The only exception is ADVAE with $\beta = 0.3$ for Wiki, but it concentrates all syntactic roles in a single latent variable; *i.e.* $N_{\Gamma^{\text{dec}}} = 1$.

sem_ src	*syn_ src*	SynPG	VGVAE	QKVAE	*target*
we have destroyed the 49th armored division.	concomitant usage is not recommended.	we have destroyed the 49th armored division.	armored division hasn't destroyed.	this military force will be destroyed.	49th armored division has been destroyed .
let the fire burn and put a piece of hot iron in it.	sing a song. sing a song for boys.	don't put the fire in it burn a hot piece of iron and fire.	burn the fire. put the iron on burns.	come on fire. get a fire on it.	keep this fire going. keep a piece of hot iron on it.
they took the lunch boxes ?	have you given me your hands ?	do they boxes took the lunch ?	have they taken them your snacks ?	have you heard of some lunch ?	have they taken the lunch boxes ?
does it have a coach ?	that's a phone switcher, right ?	how does it have a coach ?	that's a coach coach, right ?	that's a warden, huh?	it has a coach, no ?
an old lady in a cemetery.	that is a bad time for a war.	there's a lady in an old cemetery.	that's an old lady in the cemetery.	this is a strange place for a woman.	there is an old lady in the cemetery.
don't be afraid.	there are still many places to go.	you don't be afraid.	there aren't be afraid to be.	there will be no need to worry.	there is no need to be afraid .
isn't there a door open ?	the machines are still good, right ?	a isn't open door there ?	the doors aren't open, right ?	the door will be open, okay?	there is a door open, right ?

Table 9.6: Syntactic sources (*syn_ src*), semantic sources (*sem_ src*), the sentences produced when using them with different models, and the corresponding correct paraphrases (*target*).

in the qualitative examples where they often swap words in the semantic source with closely related words (*e.g.* "armored division" to "military force" in line 1, or "lunch boxes" to "snacks" in line 2). We attribute this quality to the smoothness of the latent space of VAEs which places coherent alternative lexical choices in the same vicinity. The examples above also show that our model is capable of capturing and transferring various syntactic characteristics such as the passive form (line 1), the presence of subject-verb inversion (lines 3, 4, and 7), or interjections (lines 4 and 6).

9.6 Conclusion

As a continuation to the work on unsupervised disentanglement of sentence representations presented in the previous chapter, we presented in this chapter QKVAE, an unsupervised model which is designed to disentangle syntax from semantics without syntactic or semantic information (§ 9.1). Our experiments show that its latent variables effectively position sentences in the latent space according to these attributes (§ 9.2.2). Additionally, we show that QKVAE displays clear signs of disentanglement in transfer experiments (§ 9.2.3). Although the semantic transfer is moderate, syntactic transfer with

QKVAE is competitive with SynPG, one of its supervised counterparts. We also show that VGVAE, a supervised model, needs more than 50K samples to outperform QKVAE on both syntactic and semantic transfer (§ 9.2.4). Finally, we show (§ 9.3) that QKVAE moderates the shortcomings of ADVAE when it comes to syntactic role disentanglement outside of the regularly structured dataset SNLI as speculated in the previous chapter .

We plan to extend this work in three directions: i) Finding ways to bias representations of each z_l^{sem} towards semantic proto-roles (*cf.* § 6.2) instead of syntactic roles; *ii)* Applying QKVAE to non-text data since it is data agnostic (e.g. to rearrange elements of a visual landscape.); *iii)* Investigating the behavior of QKVAE on other languages. These extensions are elaborated upon in the next chapter.

Part V

Conclusion and Perspectives

Chapter 10

Conclusion and Perspectives

The objective of the work presented in this thesis is to help remedy the dire need in NLP for explainable Deep Learning techniques. It was conducted while keeping in mind the rarity of annotated data that plagues explainability. In light of these elements, we present methods that ease obtaining explainable representations with recent Deep Learning components while requiring little-to-no annotated text data. As a conclusion to this thesis, we summarize our contributions in Section 10.1, and outline a few perspective research directions which could extend our work in Section 10.2.

10.1 Summary of Contributions

In the last 3 chapters, we detailed contributions made to data-efficient explainable Deep Learning-based NLP. In Chapter 7, we have shown that the Semi-Supervised VAE framework, in the case of sentence classification, was impeded by unnecessary components, namely the Kullback-Leibler divergence in its loss, and the unobserved latent variable in its architecture. Removing these components displayed no degradation to the performance of SSVAEs, improved their speed, and made them easier to design (*i.e.* no prior to specify) and to train (*i.e.* no posterior collapse to counteract).

The remainder of our contributions pertained to inducing understandable representations without annotations through unsupervised disentanglement with VAEs and Transformers attention. First, in Chapter 8, we presented our Attention-Driven VAE (ADVAE), which is the first VAE to use Cross-Attention to encode and decode vectorial latent variables. We have shown that these vectorial latent variables are able to spontaneously align with the realizations of core syntactic roles when trained on a dataset of regularly structured sentences. The latent variables in ADVAE actively follow separate syntactic roles using Cross-Attention, and are able to separately change the realizations of syntactic roles. Subsequently in Chapter 9, we have shown that, when different latent variables are asssigned to produce keys and values in the Cross-Attention of a VAE-based Transformer decoder, the keys naturally lean towards encoding syntactic information while the values lean towards semantic information. QKVAE, the neural network we designed to verify this hypothesis, has shown clear signs of successful syntactic and semantic transfer be-

tween sentences. Moreover, we showed that a previous supervised disentanglement model needs more than 50K samples to perform better than QKVAE, which shows that QKVAE allows for a considerable gain in annotation effort for disentanglement. Looking back to syntactic role disentanglement with this last model, we finally show that its separation between syntax and semantics also allows for better scores on different datasets when it comes to separately varying the realizations of core syntactic roles in generated sentences.

10.2 Perspectives

10.2.1 Extending our Investigations to Other Languages

As is the case for the large majority of works on disentanglement in NLP, such as the ones discussed in § 4.5, the works presented in this thesis in Chapters 8 and 9 only exhibits results on English corpora. We expect extensions to other languages to be interesting, especially for free word order language such as Arabic [Bassam et al., 2014]. The richer morphology of free word order language allows swapping the positions of syntactic roles in sentences, *e.g.* from Subject-Verb-Object (SVO) to Verb-Subject-Object (VSO), while keeping the sentences grammatical. However, although word order is more flexible for these languages, a preferred word order exists and is usually largely dominant over other word orders [Song, 2014, Chapter 4].

Since our work pertains to disentanglement of syntactic roles and to disentanglement of syntax from semantics, results on free word order languages should be informative about the extent to which ADVAE and QKVAE use word order information to achieve disentanglement.

10.2.2 A Structured Latent Variable Version of QKVAE with an Account for Compositional Semantics

The last model presented in this work, QKVAE, relies on some simplifying assumptions which hinder its ability to accurately model languages. Here are the main simplifications that we think future works could address to extend our work:

1. *Syntax and Semantics are modeled as independent generative factors*: It is clearly inaccurate to model language with independent syntax and semantics. For instance, if the semantics of a sentence are expressed by a verbal predicate that subcategorizes multiple arguments, the sentence cannot be syntactically realized through only a subject and a verb. In other words, syntax and semantics in a sentence should be bound by a common *frame*[1], which is not modeled in QKVAE.

2. *Semantics is modeled as a set of independent latent variables*: Decomposing the semantics of sentences is largely understood as identifying predicates and their argu-

[1] *Frame* here, refers to frames in Frame Semantics theory [Fillmore et al., 1976]. This theory is a theory of meaning which defines for each word a semantic frame, *i.e.* a set of typed semantic arguments that may accompany the word. From a compter science perspective, semantic frames can be understood as function signatures, where the functions are linguistic predicates. Example semantic frames can be seen on the lexical database FrameNet [Ruppenhofer et al., 2016] accessible through this URL: https://framenet.icsi.berkeley.edu/fndrupal/frameIndex.

ments, which are not independent. As a matter of fact, predicates are known to perform argument *selection* by applying paradigmatic restrictions. Furthermore, some arguments to the predicates may narrow down the selection process:

(a) $[ARG_0]$ is using a *stick*.
(b) $[ARG_0]$ is using a *computer*.

For instance, while sentences (1) and (2) display the same verbal predicate, agent selection in sentence (1) is restricted to the set of animate agents while the agent of sentence (2) should normally be in *human*, a subset of *animate*. Since semantic roles are tightly linked to core syntactic roles, this flaw is also behind syntactic role co-adaptation observed in Chapter 8.

3. *Sentence-level semantics are only trained using the sentences themselves*: when rearranging the content of a sentence in a new syntactic structure, QKVAE often changes its semantics. This is not surprising since QKVAE does not feature design choices that are targeted at well estimating, an thus, preserving sentence-level semantics (or *compositional* semantics in general). More specifically, sentence representations are not trained using the context in which sentences appear.

Future works could improve upon the above simplifications by investigating the solutions below:

Binding all latent variables to a common underlying frame In QKVAE, $p(z) = p(z^{syn}) \prod_i p(z_i^{sem})$. To deal with the first two simplifications listed above, one could introduce a *frame* variable z^f to the probabilistic model as follows $p(z) = p(z^{syn}|z^f) \prod_i p(z_i^{sem}|z^f)$. This variable would be able to conditionally bind syntax z^{syn} to semantics z^{sem}, but also the different component of z^{sem} so as to absorb syntactic role co-adaptation into this minimalistic structured latent variable model. As shown in 8.4.2, hierarchical versions of our Cross-Attention-based models are not trivial to train. In that regard, the recently introduced DELLA [Hu et al., 2022], a successfully trained structured latent variable Transformer for language modeling, should help guide the design we describe here.

Training for effective unsupervised estimation of sentence-level semantics The work of Kiros et al. [2015] on Skip-Thought vectors has shown that sentence level semantic vectors can be effectively estimated, without supervision, simply through next sentence prediction. As discussed in Chapter 5, this method continued to thrive for sentence-level representation learning over the following years and propagated to BERT [Devlin et al., 2019] and other Bert-like models. Using our semantic variable z^{sem} and our newly defined frame variable z^f, one could design a sentence representation process which takes inspiration in how syntagms compose the meaning of a sentence according to the compositional semantics governing the sentence at hand[2]. This linguistically inspired sentence-level

[2]According to our preliminary investigations, Generative Lexicon theory [Pustejovsky, 1998] constitutes a reasonable candidate to inspire such work.

representation, together with the well established next sentence prediction method could make for a method to obtain semantic representations which *i)* better model sentence-level semantics since they make use of next sentence prediction; *ii)* are more understandable since they feature a variable z^f modeling the semantic frame, other variables z_i^{sem} modeling the arguments to this frame, and linguistically interpretable interactions between these variables.

10.2.3 Applying this Work to Multi-Modal Data

As highlighted in the conclusion of Chapter 9, QKVAE is input agnostic, and could therefore as well be used for non-linguistic data. We think an interesting research direction would be to investigate the behavior of QKVAE on other modalities so as to observe the information that will be disentangled analogously to the linguistic concepts studied in our work. The idea of applying linguistically-inspired analysis to all types of data is the basis for Semiotics[3], a discipline which studies signs and symbols of all natures. In a similar fashion, the research presented in this thesis could lead, for instance, to models that could learn to extract the core syntagms[4] in pictures, regenerate the picture while changing only one of these syntagms, or produce a version of this picture where these syntagms are rearranged. As a matter of fact, the work of Jaegle et al. [2022] shows that Cross-Attention can be used to build an encoder-decoder model that can generalize to any input format with good scaling and generalization performance, and the work of Locatello et al. [2020c] shows that Cross-Attention based architectures are capable of unsupervised object segmentation on images, which encourages pursuing this research direction.

Further down the road, one could investigate our models in the multi-modal context, where multiple information channels are aggregated in order to model an event syntactically and semantically as one would model the sentence describing the event.

[3]Readers may refer to Chandler [1994] for a beginner friendly introduction to Semiotics.

[4]The term Syntagm, here, refers to an element in the structure of an observation, where the observation may or may not be textual.

References

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for mt evaluation with improved correlation with human judgments. In *Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, Proceedings of the Workshop ACL 2005*, pages 65–72, 2005. URL https://aclanthology.org/W05-0909/.

Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xinyu Dai, and Jiajun Chen. Generating Sentences from Disentangled Syntactic and Semantic Spaces. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 6008–6019, 7 2019. doi: 10.18653/v1/p19-1602. URL http://arxiv.org/abs/1907.05789.

Hammo Bassam, Moubaiddin Asma, Obeid Nadim, and Tuffaha Abeer. Formal description of arabic syntactic structure in the framework of the government and binding theory. *Computación y Sistemas*, 18(3):611–625, 2014.

Melika Behjati and James Henderson. Inducing Meaningful Units from Character Sequences with Slot Attention. *Arxiv*, 2021. URL http://arxiv.org/abs/2102.01223.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. doi: 10.1109/TPAMI.2013.50.

Alena Böhmová, Jan Hajic, Eva Hajicová, Barbora Hladká, and Anne Abeillé. The prague dependency treebank: Three-level annotation scenario. *Treebanks: building and using parsed corpora*, 20:103–127, 2003.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051. URL https://aclanthology.org/Q17-1010.

Claire Bonial, Jena Hwang, Julia Bonn, Kathryn Conger, Olga Babko-Malaya, and Martha Palmer. English propbank annotation guidelines. *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder*, 48, 2012.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL https://www.aclweb.org/anthology/D15-1075.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *CoNLL 2016 - 20th SIGNLL Conference on Computational Natural Language Learning, Proceedings*, pages 10–21, 2016. ISBN 9781945626197. doi: 10.18653/v1/k16-1002. URL http://arxiv.org/abs/1511.06349.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pages 1–14, 2016.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in-VAE. In *2017 NeurIPS Workshop on Learning Disentangled Representations*, 2017. URL http://arxiv.org/abs/1804.03599.

Daniel Chandler. *Semiotics for Beginners*. online, 1994. http://visual-memory.co.uk/daniel/Documents/S4B/(visited 2022-02-02).

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1082. URL https://aclanthology.org/D14-1082.

Jiaao Chen, Zichao Yang, and Diyi Yang. MixText: Linguistically-Informed Interpolation of Hidden Space for Semi-Supervised Text Classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157, 2020. doi: 10.18653/v1/2020.acl-main.194. URL https://www.aclweb.org/anthology/2020.acl-main.194/.

Mingda Chen, Qingming Tang, Karen Livescu, and Kevin Gimpel. Variational sequential labelers for semi-supervised learning. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 215–226, 2018a. doi: 10.18653/v1/d18-1020.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. A multi-task approach for disentangling syntax and semantics in sentence representations. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:2453–2464, 2019a. doi: 10.18653/v1/n19-1254. URL http://arxiv.org/abs/1904.01173.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984, Florence, Italy, July 2019b. Association for Computational Linguistics. doi: 10.18653/v1/P19-1599. URL https://aclanthology.org/P19-1599.

Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in vaes. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2615–2625, 2018b.

Tian Qi Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, 2018c.

Pengyu Cheng, Martin Renqiang Min, Dinghan Shen, Christopher Malon, Yizhe Zhang, Yitong Li, and Lawrence Carin. Improving Disentangled Text Representation Learning with Information-Theoretic Guidance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7530–7541, 2020. doi: 10.18653/v1/2020.acl-main.673.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014a. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012. URL https://aclanthology.org/W14-4012.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014b.

Noam Chomsky. *Syntactic Structures*. The Hague Mouton, 1957.

Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1012. URL https://aclanthology.org/N16-1012.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent nn: First results. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

Narayan Choudhary. Cost analysis of linguistic resources. *Central Institute of Indian Languages, Mysuru*, 2018.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://aclanthology.org/W19-4828.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Pre-training transformers as energy-based cloze models. In *EMNLP*, 2020. URL https://www.aclweb.org/anthology/2020.emnlp-main.20.pdf.

Pierre Colombo, Pablo Piantanida, and Chloé Clavel. A novel estimator of mutual information for learning to disentangle textual representations. *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 6539–6550, 2021. doi: 10.18653/v1/2021.acl-long.511.

Pierre Colombo, Guillaume Staerman, Nathan Noiry, and Pablo Piantanida. Learning disentangled textual representations via statistical measures of similarity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2614–2630, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.187. URL https://aclanthology.org/2022.acl-long.187.

Caio Corro and Ivan Titov. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–24, 2019.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 2978–2988, 2020. doi: 10.18653/v1/p19-1285.

Marie-Catherine De Marneffe and Christopher D Manning. Stanford typed dependencies manual. Technical report, Technical report, Stanford University, 2008.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, May 2006. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2006/pdf/440_pdf.pdf.

Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-3348. URL https://aclanthology.org/W14-3348.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1129. URL https://aclanthology.org/P14-1129.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022.

Wenyu Du, Zhouhan Lin, Yikang Shen, Timothy J O'Donnell, Yoshua Bengio, and Yue Zhang. Exploiting Syntactic Structure for Better Language Modeling: A Syntactic Distance Approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 6611–6628, 2020. doi: 10.18653/v1/2020.acl-main.591. URL https://www.aclweb.org/anthology/2020.acl-main.591/.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. Recurrent neural network grammars. *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*, pages 199–209, 2016. doi: 10.18653/v1/n16-1024.

Cian Eastwood and Christopher K.I. Williams. A framework for the quantitative evaluation of disentangled representations. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–15, 2018.

Bryan Eikema and Wilker Aziz. Is MAP decoding all you need? the inadequacy of the mode in neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4506–4520, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.398. URL https://aclanthology.org/2020.coling-main.398.

Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. A Brief Review of Domain Adaptation. In *Advances in Data Science and Information Engineering*, pages 877–894, 2021. doi: 10.1007/978-3-030-71704-9{_}65. URL https://link.springer.com/10.1007/978-3-030-71704-9_65.

Mohsen Fayyaz, Ehsan Aghazadeh, Ali Modarressi, Hosein Mohebbi, and Mohammad Taher Pilehvar. Not all models localize linguistic knowledge in the same place: A layer-wise probing on BERToids' representations. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 375–388, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.blackboxnlp-1.29. URL https://aclanthology.org/2021.blackboxnlp-1.29.

Ghazi Felhi, Joseph Le Roux, and Djamé Seddah. Challenging the semi-supervised VAE framework for text classification. In *Proceedings of the Second Workshop on Insights from Negative Results in NLP*, pages 136–143, Online and Punta Cana, Dominican Republic, November 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.insights-1.19. URL https://aclanthology.org/2021.insights-1.19.

Ghazi Felhi, Joseph Le Roux, and Djamé Seddah. Towards Unsupervised Content Disentanglement in Sentence Representations via Syntactic Roles. In *1st CtrlGen: Controllable Generative Modeling in Language and Vision Workshop at NeurIPS 2021*, 2021b. URL https://ctrlgenworkshop.github.io/camready/21/CameraReady/CTRLGEN(7).pdf.

Ghazi Felhi, Joseph Roux, and Djamé Seddah. Exploiting inductive bias in transformers for unsupervised disentanglement of syntax and semantics with VAEs. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5763–5776, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.423. URL https://aclanthology.org/2022.naacl-main.423.

Charles J Fillmore et al. Frame semantics and the nature of language. In *Annals of the New York Academy of Sciences: Conference on the origin and development of language and speech*, volume 280, pages 20–32. New York, 1976.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:240–250, 2019.

Richard Futrell, Ethan Wilcox, Takashi Morita, and Roger Levy. Rnns as psycholinguistic subjects: Syntactic state and grammatical dependency, 2018. URL https://arxiv.org/abs/1809.01329.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1243–1252. JMLR.org, 2017.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space, 2022. URL https://arxiv.org/abs/2203.14680.

Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational linguistics*, 28 (3):245–288, 2002.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Suchin Gururangan, Tam Dang, Dallas Card, and Noah A Smith. Variational pretraining for semi-supervised text classification. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 5880–5894, 2020. doi: 10.18653/v1/p19-1590.

Raza Habib, Soroosh Mariooryad, Matt Shannon, Eric Battenberg, R J Skerry-Ryan, Daisy Stanton, David Kao, and Tom Bagby. Semi-Supervised Generative Modeling for Controllable Speech Synthesis. In *International Conference on Learning Representations*, pages 1–13, 2019. URL https://iclr.cc/virtual_2020/poster_rJeqeCEtvH.html.

Sooji Han, Jie Gao, and Fabio Ciravegna. Neural language model based training data augmentation for weakly supervised early rumor detection. *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2019*, pages 105–112, 2019. doi: 10.1145/3341161.3342892.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–15, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016a. doi: 10.1109/CVPR.2016.90.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016b.

Leo Hemamou, Ghazi Felhi, Jean Claude Martin, and Chloe Clavel. Slices of Attention in Asynchronous Video Job Interviews. *2019 8th International Conference on Affective Computing and Intelligent Interaction, ACII 2019*, pages 526–551, 2019. doi: 10.1109/ACII.2019.8925439.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:4129–4138, 2019a. URL https://aclanthology.org/N19-1419.

John Hewitt and Christopher D. Manning. 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, Minneapolis, Minnesota, June 2019b. Association for Computational Linguistics. doi: 10.18653/v1/N19-1419. URL https://aclanthology.org/N19-1419.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. B-VAE: Learning basic visual concepts with a constrained variational framework. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–22, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. ISSN 08997667. doi: 10.1162/neco.1997.9.8.1735.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

Tom Hosking and Mirella Lapata. Factorising meaning and form for intent-preserving paraphrasing. *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 1405–1418, 2021. doi: 10.18653/v1/2021.acl-long.112.

Tom Hosking, Hao Tang, and Mirella Lapata. Hierarchical Sketch Induction for Paraphrase Generation. *ACL-IJCNLP 2022 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 2022. URL http://arxiv.org/abs/2203.03463.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger P Levy. A Systematic assessment of syntactic generalization in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1725–1744, 2020. doi: 10.18653/v1/2020.acl-main.158. URL https://www.aclweb.org/anthology/2020.acl-main.158.

Jinyi Hu, Xiaoyuan Yi, Wenhao Li, Maosong Sun, and Xing Xie. Fuse it more deeply! a variational transformer with layer-wise latent variable inference for text generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 697–716, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.51. URL https://aclanthology.org/2022.naacl-main.51.

James Y. Huang, Kuan-Hao Huang, and Kai-Wei Chang. Disentangling Semantics and Syntax in Sentence Embeddings with Pre-trained Language Models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1372–1379, Stroudsburg, PA, USA, 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.108. URL http://arxiv.org/abs/2104.05115https://aclanthology.org/2021.naacl-main.108.

James Y. Huang, Kuan-Hao Huang, and Kai-Wei Chang. Disentangling semantics and syntax in sentence embeddings with pre-trained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1372–1379, Online, June 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.108. URL https://aclanthology.org/2021.naacl-main.108.

Kuan-hao Huang and Kai-wei Chang. Generating Syntactically Controlled Paraphrases without Using Annotated Parallel Pairs. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1022–1033, Stroudsburg, PA, USA, 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.88. URL https://aclanthology.org/2021.eacl-main.88.

Kuan-Hao Huang and Kai-Wei Chang. Generating syntactically controlled paraphrases without using annotated parallel pairs. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1022–1033, Online, April 2021b. Association for Computational Linguistics. URL https://aclanthology.org/2021.eacl-main.88.

Zhicheng Huang, Zhaoyang Zeng, Yupan Huang, Bei Liu, Dongmei Fu, and Jianlong Fu. Seeing out of the box: End-to-end pre-training for vision-language representation learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021c.

Drew A. Hudson and Christopher D. Manning. Compositional Attention Networks for Machine Reasoning. *International Conference on Learning Representations*, 333(6045):975–8, 3 2018. ISSN 1095-9203. doi: 10.1126/science.1204534. URL http://arxiv.org/abs/1803.03067http://academic.oup.com/humupd/article/18/1/73/853086/0varian-antral-folliculogenesis-during-the-humanhttp://www.ncbi.nlm.nih.gov/pubmed/21852490.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J Henaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=fILj7WpI-g.

Sarthak Jain and Byron C. Wallace. Attention is not Explanation. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 11–20, 2 2019. doi: 10.18653/v1/d19-1002. URL http://arxiv.org/abs/1908.04626http://arxiv.org/abs/1902.10186.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–13, 2017.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy, July 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1356. URL https://aclanthology.org/P19-1356.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, 2019b. URL https://github.com/ganeshjawahar/.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1041. URL https://aclanthology.org/P19-1041.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. Disentangled representation learning for non-parallel text style transfer. In *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 424–434, 2020. ISBN 9781950737482. doi: 10.18653/v1/p19-1041.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Shivram Dattatray Joshi, Jouthe AF Roodbergen, et al. *The Aáźčáźŋādhyāyī of PāáźĞini with translation and explanatory notes*, volume 1. Sahitya Akademi, 1991.

Dan Jurafsky and James H. Martin. *Speech and Language Processing (3rd ed. draft)*. online, 2022. https://web.stanford.edu/~jurafsky/slp3/(visited 2022-12-08).

Moussa Kamal Eddine, Guokan Shang, Antoine Tixier, and Michalis Vazirgiannis. FrugalScore: Learning cheaper, lighter and faster evaluation metrics for automatic text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1305–1318, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.93. URL https://aclanthology.org/2022.acl-long.93.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *35th International Conference on Machine Learning, ICML 2018*, 6:4153–4171, 2018.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2678–2687. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/kim18e.html.

Yoon Kim, Alexander M Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. Unsupervised recurrent neural network grammars. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:1105–1117, 2019. doi: 10.18653/v1/n19-1114.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. ISBN 9781450300728. doi: 10.1145/1830483.1830503. URL http://arxiv.org/abs/1412.6980.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, pages 1–14, 2014.

Diederik P Kingma, Danilo J Rezende, Shakir Mohamed, and Max Welling. Semi-Supervised Learning with Deep Generative Models. *Advances in Neural Information Processing Systems*, 4(January):3581–3589, 6 2014. ISSN 10495258. URL http://arxiv.org/abs/1406.5298.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29:4743–4751, 2016. ISSN 10495258.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. *Advances in neural information processing systems*, 28, 2015.

Jordan Kodner and Nitish Gupta. Overestimation of syntactic representation in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 1757–1762, 2020. doi: 10.18653/v1/2020.acl-main.160. URL https://www.aclweb.org/anthology/2020.acl-main.160.

Artur Kulmizev, Vinit Ravishankar, Mostafa Abdou, and Joakim Nivre. Do neural language models show preferences for syntactic formalisms? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 4077–4091, 2020. doi: 10.18653/v1/2020.acl-main.375. URL https://www.aclweb.org/anthology/2020.acl-main.375%0A.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

Shankar Kumar and William Byrne. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 169–176, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics. URL https://aclanthology.org/N04-1022.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Joel Lang and Mirella Lapata. Unsupervised induction of semantic roles. *NAACL HLT 2010 - Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, pages 939–947, 2010.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Stroudsburg, PA, USA, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://www.aclweb.org/anthology/2020.acl-main.703.

Bohan Li, Junxian He, Graham Neubig, Taylor Berg-Kirkpatrick, and Yiming Yang. A surprisingly effective fix for deep latent variable modeling of text. In *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 3603–3614, 2020a. ISBN 9781950737901. URL https://github.com/haofuml/cyclic_.

Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 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, Stroudsburg, PA, USA, 2020b. Association for Computational Linguistics. ISBN 9781952148606. doi: 10.18653/v1/2020.emnlp-main.378. URL https://www.aclweb.org/anthology/2020.emnlp-main.378.

Dingcheng Li, Hongliang Fei, Shaogang Ren, and Ping Li. A Deep Decomposable Model for Disentangling Syntax and Semantics in Sentence Representation. *Findings of the Association for Computational Linguistics, Findings of ACL: EMNLP 2021*, pages 4300–4310, 2021a. doi: 10.18653/v1/2021.findings-emnlp.364.

Juncen Li, Robin Jia, He He, and Percy Liang. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1169. URL https://www.aclweb.org/anthology/N18-1169.

Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Deepak Gotmare, Shafiq Joty, Caiming Xiong, and Steven Hoi. Align before fuse: Vision and language representation learning with momentum distillation. In *NeurIPS*, 2021b.

Ruizhe Li, Xiao Li, Chenghua Lin, Matthew Collinson, and Rui Mao. A stable variational autoencoder for text modelling. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 594–599, Tokyo, Japan, October–November 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-8673. URL https://aclanthology.org/W19-8673.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-lm improves controllable text generation. *ArXiv*, abs/2205.14217, 2022.

Zhiyuan Li, Jaideep Vitthal Murkute, Prashnna Kumar Gyawali, and Linwei Wang. Progressive Learning and Disentanglement of Hierarchical Representations. *International Conference on Learning Representations*, 2 2020c. ISSN 23318422. URL https://openreview.net/forum?id=SJxpsxrYPShttp://arxiv.org/abs/2002.10549.

Chin-Yew Lin. Rouge: a package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004, Barcelona, Spain*, pages 74–81, July 2004. URL https://www.microsoft.com/en-us/research/publication/rouge-a-package-for-automatic-evaluation-of-summaries/.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 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, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1112. URL https://aclanthology.org/N19-1112.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Ro{bert}a: A robustly optimized {bert} pretraining approach, 2020. URL https://openreview.net/forum?id=SyxS0T4tvS.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. A commentary on the unsupervised learning of disentangled representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13681–13684, 2020a.

Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-supervised disentanglement without compromises. In *International Conference on Machine Learning*, pages 6348–6359. PMLR, 2020b.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 2020-Decem(NeurIPS):1–27, 2020c. ISSN 10495258.

Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5962–5971, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1598. URL https://aclanthology.org/P19-1598.

Guy Lorberbom, Andreea Gane, Tommi Jaakkola, and Tamir Hazan. Direct optimization through argmax for discrete variational auto-encoder. *Advances in neural information processing systems*, 32, 2019.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a. URL https://proceedings.neurips.cc/paper/2019/file/c74d97b01eae257e44aa9d5bade97baf-Paper.pdf.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019b. URL https://proceedings.neurips.cc/paper/2019/file/c74d97b01eae257e44aa9d5bade97baf-Paper.pdf.

Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2019c. ISSN 15582191. doi: 10.1109/TKDE.2018.2876857.

Minh Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015. doi: 10.18653/v1/d15-1166.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. *ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 1:142–150, 2011. URL https://www.aclweb.org/anthology/P11-1015.

Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Using Large Corpora*, 273, 1994.

Héctor Martínez Alonso, Djamé Seddah, and Benoît Sagot. From noisy questions to Minecraft texts: Annotation challenges in extreme syntax scenario. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 13–23, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://aclanthology.org/W16-3905.

Rebecca Marvin and Tal Linzen. Targeted syntactic evaluation of language models. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 1192–1202, 2020. doi: 10.18653/v1/d18-1151.

Vaden Masrani, Tuan Anh Le, and Frank Wood. The Thermodynamic Variational Objective. *Advances in Neural Information Processing Systems*, pages 11525–11534, 2019. URL https://papers.nips.cc/paper/9328-the-thermodynamic-variational-objective.pdf.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. *Advances in neural information processing systems*, 30, 2017.

Igor Aleksandrovic Mel'cuk et al. *Dependency syntax: theory and practice*. SUNY press, 1988.

Gábor Melis, Tomáš Kočiský, and Phil Blunsom. Mogrifier lstm. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJe5P6EYvS.

Sachit Menon, David Blei, and Carl Vondrick. Forget-me-not! contrastive critics for mitigating posterior collapse. In *Uncertainty in Artificial Intelligence*, pages 1360–1370. PMLR, 2022.

Giangiacomo Mercatali and André Freitas. Disentangling generative factors in natural language with discrete variational autoencoders. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3547–3556, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.301. URL https://aclanthology.org/2021.findings-emnlp.301.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing LSTM language models. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL http://arxiv.org/abs/1301.3781.

Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *The IEEE International Conference on Computer Vision (ICCV)*, Nov 2019.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://aclanthology.org/L16-1262.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://www.aclweb.org/anthology/2020.lrec-1.497.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, volume 371, pages 311–318, Morristown, NJ, USA, 2001. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL http://portal.acm.org/citation.cfm?doid=1073083.1073135.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. ISSN 10495258. doi: 10.3115/v1/D14-1162. URL http://aclweb.org/anthology/D14-1162.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge enhanced contextual word representations. 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 43–54, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1005. URL https://aclanthology.org/D19-1005.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. Information-Theoretic Probing for Linguistic Structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Stroudsburg, PA, USA, 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.420. URL http://arxiv.org/abs/2004.03061https://www.aclweb.org/anthology/2020.acl-main.420.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online, July 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.420. URL https://aclanthology.org/2020.acl-main.420.

James Pustejovsky. *The generative lexicon*. MIT press, 1998.

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

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

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Shauli Ravfogel, Yanai Elazar, Jacob Goldberger, and Yoav Goldberg. Unsupervised distillation of syntactic information from contextualized word representations. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 91–106, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.blackboxnlp-1.9. URL https://aclanthology.org/2020.blackboxnlp-1.9.

Danilo Jimenez Rezende, S. M. Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3D structure from images. In *Advances in Neural Information Processing Systems*, pages 5003–5011, 2016.

Karl Ridgeway and Michael C. Mozer. Learning deep disentangled embeddings with the F-statistic loss. *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS):185–194, 2018. ISSN 10495258.

Geoffrey Roeder, Yuhuai Wu, and David Duvenaud. Sticking the landing: Simple, lower-variance gradient estimators for variational inference. *Advances in Neural Information Processing Systems*, 2017-Decem: 6926–6935, 2017. ISSN 10495258.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A Primer in BERTology: What we know about how BERT works. *arXiv*, 2020a.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020b. doi: 10.1162/tacl_a_00349. URL https://aclanthology.org/2020.tacl-1.54.

Raúl Rojas. A tutorial introduction to the lambda calculus. *arXiv preprint arXiv:1503.09060*, 2015.

Michal Rolinek, Dominik Zietlow, and Georg Martius. Variational autoencoders pursue pca directions (by accident). *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:12398–12407, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.01269.

Alexey Romanov, Anna Rumshisky, Anna Rogers, and David Donahue. Adversarial decomposition of text representation. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:815–825, 2019. doi: 10.18653/v1/n19-1088.

Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16:1–80, 1 2022. ISSN 1935-7516. doi: 10.1214/21-SS133. URL http://arxiv.org/abs/2103.11251https://projecteuclid.org/journals/statistics-surveys/volume-16/issue-none/Interpretable-machine-learning-Fundamental-principles-and-10-grand-challenges/10.1214/21-SS133.full.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Josef Ruppenhofer, Michael Ellsworth, Myriam Schwarzer-Petruck, Christopher R Johnson, and Jan Scheffczyk. Framenet ii: Extended theory and practice. Technical report, International Computer Science Institute, 2016.

Amrita Saha, Ghulam Ahmed Ansari, Abhishek Laddha, Karthik Sankaranarayanan, and Soumen Chakrabarti. Complex program induction for querying knowledge bases in the absence of gold programs. *Transactions of the Association for Computational Linguistics*, 7:185–200, 2019. doi: 10.1162/tacl_a_00262. URL https://aclanthology.org/Q19-1012.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

Sashank Santhanam and Samira Shaikh. Emotional neural language generation grounded in situational contexts. In *Proceedings of the 4th Workshop on Computational Creativity in Language Generation*, pages 22–27, Tokyo, Japan, 29 October–3 November 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2019.ccnlg-1.3.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992. doi: 10.1162/neco.1992.4.1.131.

Florian Schmidt, Stephan Mandt, and Thomas Hofmann. Autoregressive text generation beyond feedback loops. 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 3400–3406, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1338. URL https://aclanthology.org/D19-1338.

Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, 2012. doi: 10.1109/ICASSP.2012.6289079.

Djamé Seddah, Farah Essaidi, Amal Fethi, Matthieu Futeral, Benjamin Muller, Pedro Javier Ortiz Suárez, Benoît Sagot, and Abhishek Srivastava. Building a User-Generated Content North-African Arabizi Treebank: Tackling Hell. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1139–1150. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.107. URL https://www.aclweb.org/anthology/2020.acl-main.107.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2074. URL https://aclanthology.org/N18-2074.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. *35th International Conference on Machine Learning, ICML 2018*, 10:7322–7330, 2018.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *7th International Conference on Learning Representations, ICLR 2019*, pages 1–14, 2019.

Yikang Shen, Shawn Tan, Alessandro Sordoni, Peng Li, Jie Zhou, and Aaron Courville. Unsupervised dependency graph network. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4767–4784, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.327. URL https://aclanthology.org/2022.acl-long.327.

Stuart M Shieber. Evidence against the context-freeness of natural language. In *Philosophy, language, and artificial intelligence*, pages 79–89. Springer, 1985.

Aditya Siddhant, Melvin Johnson, Henry Tsai, Naveen Arivazhagan, Jason Riesa, Ankur Bapna, Orhan Firat, and Karthik Raman. Evaluating the cross-lingual effectiveness of massively multilingual neural machine translation, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model, 2022. URL https://arxiv.org/abs/2201.11990.

Jae Jung Song. *Linguistic typology: Morphology and syntax*. Routledge, 2014.

Milan Straka. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/K18-2020. URL https://www.aclweb.org/anthology/K18-2020.

Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, Lei Hou, Maosong Sun, and Jie Zhou. On transferability of prompt tuning for natural language processing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3949–3969, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.290. URL https://aclanthology.org/2022.naacl-main.290.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England, August 2008. Coling 2008 Organizing Committee. URL https://aclanthology.org/W08-2121.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, 155(1-2):105–145, 9 2014. ISSN 0025-5610. doi: 10.5555/2969033.2969173. URL http://arxiv.org/abs/1409.3215http://link.springer.com/10.1007/s10107-014-0839-0.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 4593–4601, 2020. doi: 10.18653/v1/p19-1452. URL http://arxiv.org/abs/1905.05950.

Lucien Tesnière. *Éléments de syntaxe structurale*. Klincksieck, 1959.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

Andros Tjandra, Ruoming Pang, Yu Zhang, and Shigeki Karita. Unsupervised learning of disentangled speech content and style representation. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 4:3191–3195, 2021. ISSN 19909772. doi: 10.21437/Interspeech.2021-1936.

Jannis Vamvas and Rico Sennrich. On the Limits of Minimal Pairs in Contrastive Evaluation. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 58–68, Stroudsburg, PA, USA, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.blackboxnlp-1.5. URL http://arxiv.org/abs/2109.07465https://aclanthology.org/2021.blackboxnlp-1.5.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446.

Yunli Wang, Yu Wu, Lili Mou, Zhoujun Li, and Wenhan Chao. Harnessing pre-trained neural networks with rules for formality style transfer. 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 3573–3578, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1365. URL https://aclanthology.org/D19-1365.

Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. NLProlog: Reasoning with weak unification for question answering in natural language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6151–6161, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1618. URL https://aclanthology.org/P19-1618.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. OntoNotes Release 5.0, 2013. URL https://hdl.handle.net/11272.1/AB2/MKJJ2R.

Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 11–20, 2020. doi: 10.18653/v1/d19-1002.

John Wieting and Kevin Gimpel. ParanMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 1:451–462, 2018. doi: 10.18653/v1/p18-1042.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. What do RNN language models learn about filler–gap dependencies? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 211–221, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5423. URL https://aclanthology.org/W18-5423.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. Structural supervision improves learning of non-local grammatical dependencies. 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 3302–3312, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1334. URL https://aclanthology.org/N19-1334.

Lawrence Wolf-Sonkin, Jason Naradowsky, Sebastian J Mielke, and Ryan Cotterell. A structured variational autoencoder for contextual morphological inflection. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 1:2631–2641, 2018. doi: 10.18653/v1/p18-1245.

Chen Wu, Prince Zizhuang Wang, and William Yang Wang. On the encoder-decoder incompatibility in variational text modeling and beyond. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3464, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.316. URL https://aclanthology.org/2020.acl-main.316.

Linjuan Wu, Shaojuan Wu, Xiaowang Zhang, Deyi Xiong, Shizhan Chen, Zhiqiang Zhuang, and Zhiyong Feng. Learning disentangled semantic representations for zero-shot cross-lingual transfer in multilingual machine reading comprehension. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 991–1000, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.70. URL https://aclanthology.org/2022.acl-long.70.

Peng Xu, Jackie Chi Kit Cheung, and Yanshuai Cao. On variational learning of controllable representations for text without supervision. In *International Conference on Machine Learning*, pages 10534–10543. PMLR, 2020a.

Peng Xu, Jackie Chi Kit Cheung, and Yanshuai Cao. On Variational Learning of Controllable Representations for Text without Supervision. *The 37th International Conference on Machine Learning (ICML 2020)*, 2020b. URL http://arxiv.org/abs/1905.11975.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. Variational autoencoder for semi-supervised text classification. *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 3358–3364, 2017.

Yaniv Yacoby, Weiwei Pan, and Finale Doshi-Velez. Failure Modes of Variational Autoencoders and Their Effects on Downstream Tasks. In *ICML Workshop on Uncertainty {\&} Robustness in Deep Learning (UDL)*, 2020. URL http://arxiv.org/abs/2007.07124.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. Breaking the softmax bottleneck: A high-rank rnn language model. In *International Conference on Learning Representations*, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical Attention Networks for Document Classification. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016. ISSN 1606.02393. doi: 10.18653/v1/N16-1174. URL http://aclweb.org/anthology/N16-1174.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 2852–2858, 2017.

Cheng Zhang, Judith Butepage, Hedvig Kjellstrom, and Stephan Mandt. Advances in Variational Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, 2019a. ISSN 19393539. doi: 10.1109/TPAMI.2018.2889774.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1205. URL https://aclanthology.org/P18-1205.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657, 2015.

Xinyuan Zhang, Yi Yang, Siyang Yuan, Dinghan Shen, and Lawrence Carin. Syntax-Infused Variational Autoencoder for Text Generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2069–2078, Stroudsburg, PA, USA, 6 2019b. Association for Computational Linguistics. ISBN 9781950737482. doi: 10.18653/v1/P19-1199. URL http://arxiv.org/abs/1906.02181https://www.aclweb.org/anthology/P19-1199.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models. *CoRR*, abs/1702.08658, 2017a. URL http://arxiv.org/abs/1702.08658.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Learning hierarchical features from deep generative models. *34th International Conference on Machine Learning, ICML 2017*, 8:6195–6204, 2017b.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658*, 2017c.

Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, Los Alamitos, CA, USA, dec 2015. IEEE Computer Society. doi: 10.1109/ICCV.2015.11. URL https://doi.ieeecomputersociety.org/10.1109/ICCV.2015.11.

Appendix A

Background

A.1 Derivations for the TC-VAE decomposition

Here, we lay out the derivations necessary to obtain the decomposition displayed in Eq. 4.2 in Section 4.1. We start from the 3 terms on the left-hand-side:

$$KL[q_\phi(z,n)||q_\phi(z)p(n)] + KL[q_\phi(z)||\prod_j q_\phi(z_j)] + \sum_j KL[q_\phi(z_j)||p(z_j)] \tag{A.1}$$

$$\begin{aligned}
=&\mathbb{E}_{(z;n)\sim q_\phi(z,n)}\left[\log \frac{q_\phi(z,n)}{q_\phi(z)p(n)}\right] + \mathbb{E}_{z\sim q_\phi(z)}\left[\log \frac{q_\phi(z)}{\prod_j q_\phi(z_j)}\right] \\
&+ \sum_j \mathbb{E}_{z_j\sim q_\phi(z_j)}\left[\log \frac{q_\phi(z_j)}{p(z_j)}\right]
\end{aligned} \tag{A.2}$$

$$\begin{aligned}
=&\sum_n \int_z q_\phi(z,n)\log\frac{q_\phi(z,n)}{q_\phi(z)p(n)}dz + \int_z q_\phi(z)\log\frac{q_\phi(z)}{\prod_j q_\phi(z_j)}dz \\
&+ \sum_j \int_{z_j} q_\phi(z_j)\log\frac{q_\phi(z_j)}{p(z_j)}dz_j
\end{aligned} \tag{A.3}$$

We first unify all three terms under the expectation over $q_\phi(z,n)$. For the second term we can use $q_\phi(z) = \sum_n q_\phi(z,n)$:

$$\int_z q_\phi(z)\log\frac{q_\phi(z)}{\prod_j q_\phi(z_j)}dz = \sum_n \int_z q_\phi(z,n)\log\frac{q_\phi(z)}{\prod_j q_\phi(z_j)}dz \tag{A.4}$$

For the third term, we apply an expectation over $q_\phi(z_1,\ldots,z_{j-1},z_{j+1},\ldots,z_{|z|}|z_j)$ on each element of the sum over j [1], then also use $q_\phi(z) = \sum_n q_\phi(z,n)$:

[1] The expectation of an expression over the distribution of variables which are not involved in the expression is the expression itself, *i.e.* $\mathbb{E}_{x\sim p(x)}[f(y)] = f(y)$.

$$\sum_j \int_{z_j} q_\phi(z_j) \log \frac{q_\phi(z_j)}{p(z_j)} dz_j \tag{A.5}$$

$$= \sum_j \int_{z_1,\ldots,z_{j-1},z_{j+1},\ldots,z_{|z|}} \int_{z_j}$$

$$q_\phi(z_1,\ldots,z_{j-1},z_{j+1},\ldots,z_{|z|}|z_j) q_\phi(z_j) \log \frac{q_\phi(z_j)}{p(z_j)} dz_j dz_1,\ldots,dz_{j-1},dz_{j+1},\ldots,dz_{|z|} \tag{A.6}$$

$$= \sum_j \int_z q_\phi(z) \log \frac{q_\phi(z_j)}{p(z_j)} dz \tag{A.7}$$

$$= \int_z q_\phi(z) \log \frac{\prod_j q_\phi(z_j)}{\prod_j p(z_j)} \tag{A.8}$$

$$= \sum_n \int_z q_\phi(z,n) \log \frac{\prod_j q_\phi(z_j)}{\prod_j p(z_j)} \tag{A.9}$$

Finally, fusing the three terms under the same expectation yields:

$$\mathbb{E}_{(z;n)\sim q_\phi(z,n)} \left[\log \frac{q_\phi(z,n)}{q_\phi(z)p(n)} + \log \frac{q_\phi(z)}{\prod_j q_\phi(z_j)} + \log \frac{\prod_j q_\phi(z_j)}{\prod_j p(z_j)} \right] \tag{A.10}$$

$$= \mathbb{E}_{(z;n)\sim q_\phi(z,n)} \left[\log \frac{q_\phi(z,n)}{\cancel{q_\phi(z)}p(n)} \frac{\cancel{q_\phi(z)}}{\cancel{\prod_j q_\phi(z_j)}} \frac{\cancel{\prod_j q_\phi(z_j)}}{\prod_j p(z_j)} \right] \tag{A.11}$$

Since $\frac{q_\phi(z,n)}{p(n)} = q_\phi(z|n)$ and $p(z)$ has independent components (*i.e.* $p(z) = \prod_j p(z_j)$), the derivation continues as follows:

$$\mathbb{E}_{(z;n)\sim q_\phi(z,n)} \left[\log \frac{q_\phi(z|n)}{p(z)} \right] = \mathrm{KL}\left[q_\phi(z|n) || p(z) \right] \tag{A.12}$$

which proves the equality in Equation 4.2.

Appendix B

Unsupervised Disentanglement of Syntactic Roles

This appendix contains the supplementary materials for our work on unsupervised disentanglement of syntactic roles. In section B.1, we measure the effect of varying latent variables on the appearance/disappearance of syntactic roles to analyze the influence of these latent variables on the *structure* of sentences rather than their content. Appendix B.2 displays a few sentences from both SNLI and Yelp datasets together with the syntactic role extractions obtained with our extraction heuristic to get an idea of the successes/failures of this heuristic. In Appendix B.3, we provide the training details and hyper-parameters for our experiments. In Appendix B.4 we display the fine-grained syntactic role-wise disentanglement scores corresponding to the global scores we display in the core text. Appendix B.5 shows influence heatmaps produced by our model when measured for a wide range of syntactic roles over Stanford Dependency-type annotations and Universal Dependency-type annotations, and also for PoS tags. In Appendix B.6 we give a larger array of random examples demonstrating the controlled generation on syntactic roles realizations enabled by ADVAE. Appendix B.7 provides standard VAE language modeling metrics reported for the runs we conducted with the different architectures we compare. Since for our encoder-related metrics we average attention values over all the layers, we display Appendix B.8 layer-wise encoder influence heatmaps to show that that trends we described can also be observed on individual layers. Finally, Appendix B.9 display disentanglement results for ADVAE over a larger grid of values for L, the number of vectorial latent variables used.

B.1 Measuring the effect of latent variables on the structure of sentences

In Figure B.1, for each latent variable and each syntactic role, we report the probability that resampling the latent variable causes the appearance/disappearance of the syntactic role. The instance we use here is the same as the one we use for the heatmaps in the main body of the paper. According to the heatmaps in Figures 8.3 and 8.4, latent variable 3 is the one associated with the verb. As can be seen in the present heatmap in Figure B.1, this same variable is the one that has the most influence on the appearance/disappearance of direct and prepositional objects, and this is a pattern that proved to be consistent across

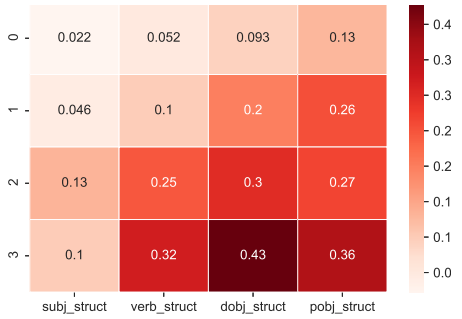


Figure B.1: The influence of latent variables on the appearance or disappearance of syntactic roles.

our different runs. This constitutes empirical justification for our choice of discarding these cases from our decoder influence metrics.

B.2 Example Sentences from Yelp and SNLI and their Corresponding Syntactic Extractions

Table B.1 shows some samples from SNLI and Yelp reviews. Samples from Yelp Reviews exhibit a clearly higher structural diversity. On the other hand, most SNLI samples are highly similar in structure.
Our syntactic role extraction heuristics were tailored for sentences with verbal roots. As a result, it can be seen that they struggle with sentences with nominal roots as well as other forms of irregular utterances present in Yelp. For SNLI, our extractions mostly yield the expected results, allowing for a reliable global assessment of our models.

B.3 Training Details and Hyper-Parameter Settings

Our ADVAE's hyper-parameters Our model has been set to be large enough to reach a low reconstruction error during the initial reconstruction phase of the training. We use 2-layer Transformers with 4 attention heads and a hidden size of 192. Contrary to Vanilla VAEs, our model seems to perform better with high values of L. Therefore, we set our latent vector to a size of 768, and divide it into 96-dimensional variables for our $L = 8$ model and to 192-dimensional latent variables for our $L = 4$ model. No automated hyper-parameter selection has been done afterward.

Sequence VAE hyper-parameters As is usually done for this baseline [Xu et al., 2020b], we set both the encoder and the decoder to be 2-layer LSTMs.
We run this model for hidden LSTM sizes in [256, 512], and latent vector sizes in [16, 32]. The results for the model scoring the highest $\mathbb{D}_{dec}$ are then reported. Even though selection has been done according to $\mathbb{D}_{dec}$, we checked the remaining instances of our baselines and they also yielded low $N_{\Gamma^{\text{dec}}}$ values.

Table B.1: Example syntactic role extractions from both SNLI and Yelp

Source	Sentence	subj	verb	dobj	pobj
Yelp	i was originally told it would take _num_ mins .	it	told	_ num _ mins	
Yelp	slow , over priced , i 'll go elsewhere next time .	i	go		
Yelp	we will not be back	we			
Yelp	terrible .				
Yelp	at this point they were open and would be for another hour .	they			this point
SNLI	people are outside playing baseball .	people		baseball	
SNLI	two dogs pull on opposite ends of a rope .	two dogs	pull	opposite ends of a rope	a rope
SNLI	a lady lays at a beach .	a lady	lays		a beach
SNLI	people are running through the streets while people watch .	people	running		the streets
SNLI	someone prepares food into bowls	someone	prepares	food	bowls

Transformer VAE hyper-parameters We set the hidden sizes and number of layers for this baseline similarly to ADVAE, since it is also a Transformer. We run this model for latent vector sizes in [16, 32] and display the highest scoring model, as is done for the Sequence VAE.

Training phases All our models are trained using ADAM [Kingma and Ba, 2015] with a batch size of 128 and a learning rate of 2e-4 for 20 epochs. The dropout is set to 0.3. To avoid posterior collapse, we train all our models for 3000 steps with $\beta = 0$ (reconstruction phase), then we linearly increase β to its final value for the subsequent 3000 steps. Following Bowman et al. [2016], we also use word-dropout. We set its probability to 0.1.

Evaluation For the evaluation, T^{dec} is set to 2000, and T^{enc} is equal to the size of the test set.

B.4 Disentanglement Scores for each Syntactic Role

The full disentanglement scores are reported in Table B.2 for the decoder, and in Table B.3 for the encoder.

B.5 Disentanglement Heatmaps Over the Entire Range of Syntactic Roles and PoS Tags

We report decoder and encoder heatmaps for all the syntactic roles following the Stanford Dependencies (SD; De Marneffe and Manning, 2008) annotation scheme of Ontonotes, which was used to train our Spacy2 parser, in Figures B.2 and B.3. For the sake of ex-

Table B.2: Complete decoder disentanglement scores for SNLI

Model	β	$\mathbb{D}_{dec}$	$N_{\Gamma^{\text{dec}}}$	$\Delta\Gamma_{dec,verb}$	$\Delta\Gamma_{dec,subj}$	$\Delta\Gamma_{dec,dobj}$	$\Delta\Gamma_{dec,pobj}$
ours-4	0.3	0.78(0.10)	3.00(0.00)	0.41(0.17)	0.33(0.09)	0.03(0.01)	0.02(0.02)
	0.4	0.84(0.10)	3.00(0.00)	0.47(0.04)	0.31(0.07)	0.04(0.01)	0.01(0.01)
ours-8	0.3	0.62(0.17)	3.20(0.45)	0.32(0.08)	0.23(0.14)	0.04(0.01)	0.03(0.03)
	0.4	0.80(0.11)	3.00(0.00)	0.45(0.06)	0.27(0.05)	0.04(0.03)	0.04(0.03)
Sequence VAE	0.3	0.43(0.18)	1.70(0.48)	0.07(0.05)	0.26(0.10)	0.04(0.05)	0.06(0.05)
	0.4	0.91(0.32)	1.40(0.52)	0.24(0.13)	0.45(0.13)	0.05(0.05)	0.16(0.13)
Transformer VAE	0.3	0.08(0.04)	3.00(0.71)	0.05(0.06)	0.01(0.01)	0.01(0.01)	0.01(0.01)
	0.4	0.11(0.05)	3.80(0.45)	0.04(0.03)	0.04(0.03)	0.02(0.01)	0.01(0.02)

Table B.3: Complete encoder disentanglement scores for SNLI

Model	β	$\mathbb{D}_{enc}$	$N_{\Gamma^{\text{enc}}}$	$\Delta\Gamma_{enc,verb}$	$\Delta\Gamma_{enc,subj}$	$\Delta\Gamma_{enc,dobj}$	$\Delta\Gamma_{enc,pobj}$
ours-4	0.3	1.30(0.09)	3.00(0.00)	0.28(0.05)	0.65(0.02)	0.08(0.03)	0.29(0.03)
	0.4	1.46(0.33)	3.00(0.00)	0.38(0.12)	0.64(0.10)	0.14(0.04)	0.30(0.10)
ours-8	0.3	1.36(0.13)	3.40(0.89)	0.44(0.12)	0.60(0.18)	0.21(0.08)	0.11(0.06)
	0.4	1.44(0.79)	3.40(0.55)	0.42(0.23)	0.61(0.34)	0.17(0.10)	0.23(0.16)
Average Position	-	0.98 (-)	3.00(-)	0.12(-)	0.70(-)	0.12(-)	0.04(-)

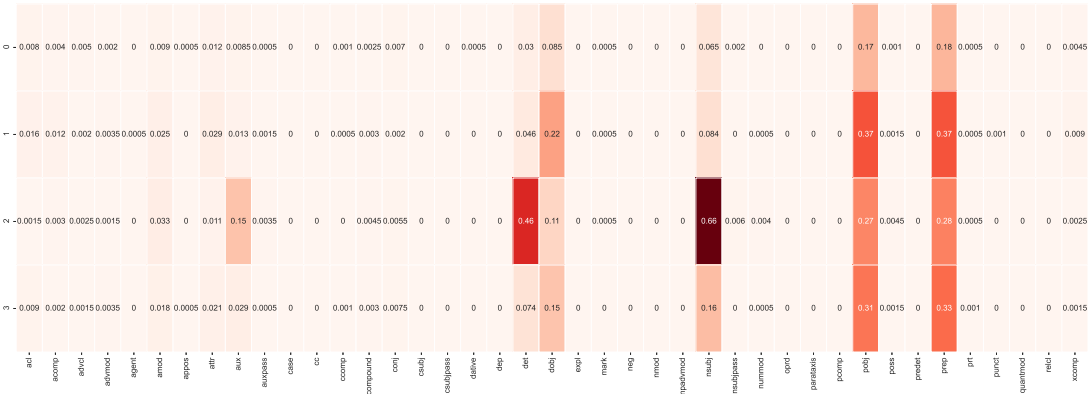


Figure B.2: Decoder influence heatmap for all SD syntactic roles.

tensiveness and to make sure we did not draw results from some parser biases, we also report the same heatmaps but using UDPipe 2.0 [Straka, 2018], which uses UD type annotations[1], in Figures B.6 and B.7. Finally, we also report heatmaps for interaction with PoS Tags extracted with Spacy2 in Figures B.4 and B.5. As was done in the main body of the paper, the span corresponding to each syntactic role (in both annotation schemes) was taken to be the series of words included in its corresponding subtree. In contrast, the span corresponding to each PoS tag was just taken to be the tagged word. Results from UD parsing extraction lead to the same conclusions as from our initial SD results.

The instance of our ADVAE for which we display the above heatmaps is the same one

[1] A widely adopted annotation scheme derived from Stanford Dependencies.

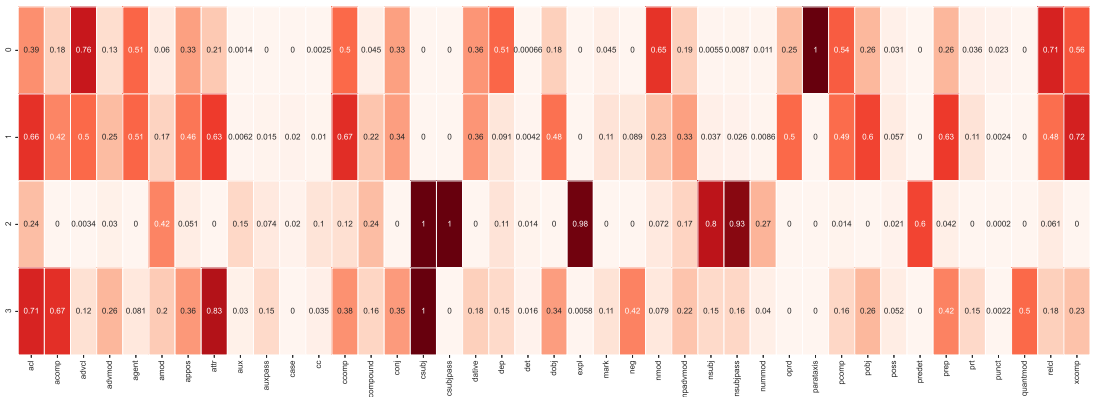


Figure B.3: Encoder influence heatmap for all SD syntactic roles.

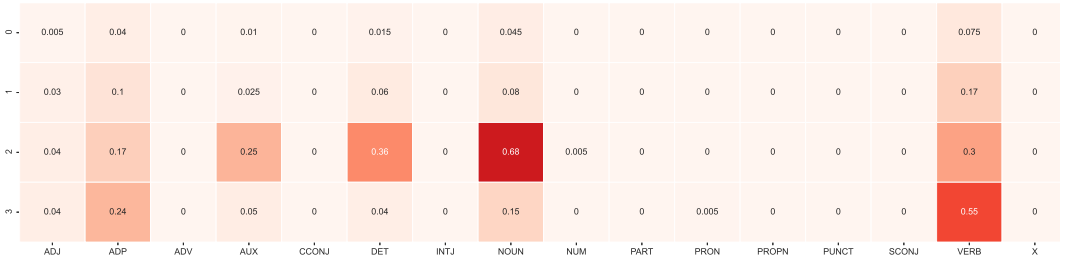


Figure B.4: Decoder influence heatmap for all PoS Tags.

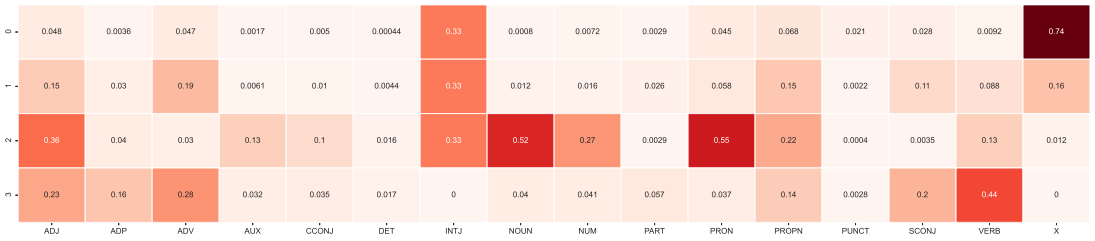


Figure B.5: Encoder influence heatmap for all PoS Tags.

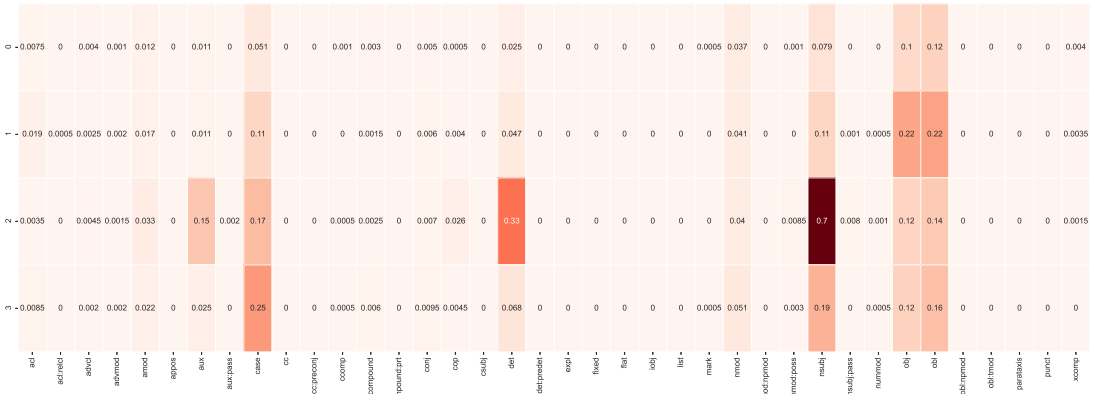


Figure B.6: Decoder influence heatmap for all UD syntactic Roles.

for which we display the heatmaps in Figures 8.3 and 8.4 in the main body of the paper.

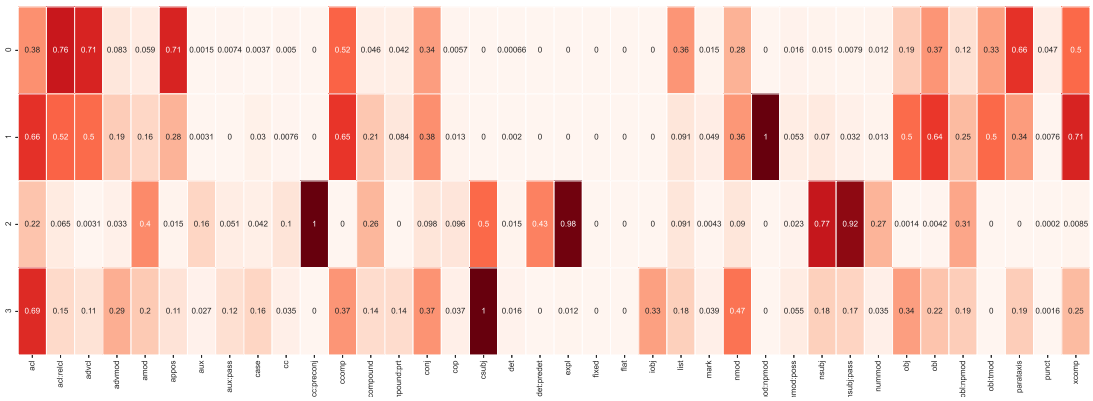


Figure B.7: Encoder influence heatmap for all UD syntactic Roles.

As shown in those Figures, it mostly uses variable 3 for verbs, variables 2 for subjects, and variable 1 for objects. The remaining variable (0) also seems to capture some interaction with objects. The heatmaps show that our ADVAE tends to group syntactic roles into latent variables in a way that aligns with the predicative structure of sentences. In fact, variable 2 displays the highest influence on the PoS tag VERB as well as its surroundings as a predicate argument such as adverbs and adverbial phrases. Similarly, latent variable 2 displays a high influence on subjects (nominal or clausal), numeral modifiers, adjectival modifiers, and auxiliaries (for conjugation). Moreover, Variable 1 highly influences the direct and prepositional objects, which we study in the main body of the paper, but also diverse clausal modifiers and obliques which often play similar roles to direct and prepositional objects in a predicate structure.

B.6 Additional Examples of Resampled realizations for each syntactic role

Table B.4 contains a wide array of examples where the latent variable corresponding to each syntactic role is resampled.

Table B.4: More examples where we resample a specific latent variable for a sentence.

Original sentence	Resampled subject	Resampled verb	Resampled dobj/pobj
the woman is riding a large brown dog	two men are riding in a large city	the woman is wet	the woman is riding on the bus
the police are running in a strategy	a man is looking at a date	the police are at an arid	the police are running in a wooded area
a man is holding a ball	a man is holding a ball	a man is , and a woman are talking on a road	a man is sitting on a cellphone outside
everyone is watching the game	some individuals are watching tv	everyone is a man	everyone is watching the game in the air
there is a man in the air	a man is sitting in the air	there is no women wearing swim trunks	there is a man in a red shirt
a group of friends are standing on a beach	an elderly father and child are standing on the beach	a group of people are standing on a beach	a group of friends are looking at the beach
the women are in a store	a man is playing a game	two women are on a break	two women are sitting on a bench
a man is playing a game	a little girl is playing with a ball	a man is clean	a man is sitting on a lake to an old country
a man is playing a game	some dogs are playing in the pool	a man is preparing to chase himself	a man is playing a game
the memorial woman is happy	a dog is happy	the memorial workers are in a room	the memorial is happy
a man is wearing a green jacket and a ship	a boy sitting in a green device	a man is dancing for the camera	the man is wearing a hat
a man is playing a game	a man is playing a game	two men are tripod	a man is playing with a guitar
a man is wearing a brown sweater and green shirt	a karate dog is swimming in a chair	a man is bought a brown cat in an airplane	a man is wearing a dress and talks to the woman
the woman is about to visitors	three people are working at a babies	the woman is wearing a sewer	the woman is about to sell a tree
a man is sitting in the snowy field	a man is sitting in the snowy field	a man is wearing electronics	a man is sitting on a park bench
two people are playing in the snow	the motorcycle is a woman on the floor	two people play soccer in the snow	two people are playing in a concert
a man is standing next to another man	a boy is standing next to another man	a man is standing	a man is standing next to a man
a man is on his bike	a man is on his bike	a dog is showing water	a man is on his bike
a man is sitting in front of a tree , taking a picture	a man is sitting in front of a tree	a man is holding a red shirt and climbing a tree	a man is sitting on a suburban own
a man is sitting with a dog	the children are sitting with the dog	a man is playing with a dog	a man is sitting with an umbrella

B.7 Reconstruction and Kullback-Leibler Values Across Experiments

Table B.5: Reconstruction loss and Kullback-Leibler values on SNLI.

Model	β	$-\mathbb{E}_{(z)\sim q_\phi(z\|x)}[\log p_\theta(x\|z)]$	$\mathrm{KL}[q_\phi(z\|x)\|\|p(z)]$	Perplexity Upper Bound
Sequence VAE	0.3	31.38(0.12)	2.80(0.25)	22.02(0.30)
	0.4	32.19(0.13)	1.22(0.04)	21.08(0.22)
Transformer VAE	0.3	24.35(0.14)	13.38(0.19)	25.07(0.27)
	0.4	26.57(0.27)	8.36(0.32)	20.68(0.16)
ours-4	0.3	10.75(0.94)	42.63(1.16)	68.49(5.96)
	0.4	16.01(0.64)	27.93(1.52)	36.16(2.20)
ours-8	0.3	8.83(1.66)	46.99(2.99)	77.26(9.02)
	0.4	16.84(8.50)	27.34(14.99)	39.23(11.27)

Table B.6: Reconstruction loss and Kullback-Leibler values on Yelp.

Model	β	$-\mathbb{E}_{(z)\sim q_\phi(z\|x)}[\log p_\theta(x\|z)]$	$\mathrm{KL}[q_\phi(z\|x)\|\|p(z)]$	Perplexity Upper Bound
Sequence VAE	0.3	32.55(0.27)	4.26(0.57)	36.97(0.82)
	0.4	33.35(0.11)	1.42(0.15)	32.42(0.13)
Transformer VAE	0.3	23.64(0.11)	19.24(0.32)	53.94(1.14)
	0.4	26.41(0.11)	12.79(0.20)	41.25(0.55)
ours-4	0.3	7.30(0.27)	55.19(0.30)	121.44(5.16)
	0.4	18.19(4.38)	29.85(6.52)	58.11(8.40)
ours-8	0.3	5.36(0.48)	59.24(0.61)	129.54(7.63)
	0.4	15.36(5.75)	34.16(12.40)	63.62(16.56)

The values for the reconstruction loss, the KL divergence, and the upper bound on perplexity concerning the experiments in the main body of the paper are reported in Table B.5. The same value for the Yelp experiments are in Table B.6. Since our models are VAE-based, one can only obtain the upper bound on the perplexity and not its exact value. These upper bound values are obtained using an importance sampling-based estimate of the negative log-likelihood, as was done in Wu et al. [2020]. We set the number of importance samples to 10.

It can be seen that the behavior of ADVAEs is very different from classical Sequence VAEs and Transformer VAEs. On the plus side, they are capable of sustaining much more information in their latent variables as shown by their higher KL, and they do better at reconstruction. The upper bound estimate of their perplexity is however higher. A high KL makes it more difficult for the importance sampling-based perplexity estimate to reach the true value of the model's perplexity. This may be the reason behind the higher values observed for ADVAEs.

B.8 Layer-wise Encoder Attention

In the main body of the paper, we use attention values that are averaged throughout the network. We display the encoder heatmaps obtained by using attention values from the first layer (Fig. B.8), the second layer (Fig. B.9), or an average on both layers (Fig. B.10) for comparison.

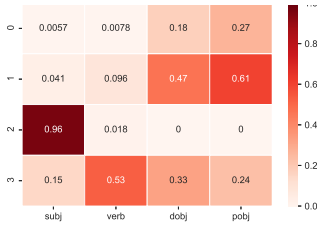


Figure B.8: Encoder influence heatmap (Γ^{enc}) when only using the *first* layer.

Figure B.9: Encoder influence heatmap (Γ^{enc}) when only using the *second* layer.

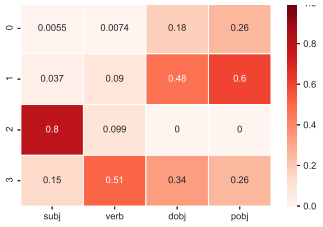


Figure B.10: Encoder influence heatmap (Γ^{enc}) when *averaging* over both layers.

As can be seen, the first layer alone provides the most sparse heatmap, and thus, the clearest correspondence between syntactic roles and latent variables. This corroborates the claims of Tenney et al. [2020] about syntax being most prominently processed in the early layers of Transformers.

B.9 ADVAE Results for a larger grid of L values

We display in Table B.7 the quantitative results of ADVAE on SNLI for L in $\{2, 4, 6, 8\}$. For ours-2, it is normal that it only separates syntactic role realizations into a maximum of 2 latent variables, as seen from the values of $N_{\Gamma^{\text{enc}}}$ and $N_{\Gamma^{\text{enc}}}$, since 2 is its total number of latent variables.

As observed in the main body of the thesis, the increase of the number of latent variables used in ADVAE leads to dispatching the influence on the realization of a single syntactic role to multiple latent variables. This in turn,leads to the decrease observed for $\mathbb{D}_{enc}$ and $\mathbb{D}_{dec}$. In Figures B.11 and B.12, we respectively display the encoder and decoder heatmaps of ADVAE with 16 latent variables. As can be seen in these figures, latent variables still specializes in specific syntactic roles. This is seen more clearly on the encoder heatmap due to co-adaptation harming the clarity of the decoder heatmap. This specialization seems to be shared among groups (*e.g.* variables 0, 7 and 8 specialize in the subject, as indicated by the green squares on the figure). This causes the difference of influence between the most influential variable and the second most influential one to be low, and thus decreases the values of $\mathbb{D}_{enc}$ and $\mathbb{D}_{dec}$.

Table B.7: Disentanglement quantitative results on SNLI for a larger grid of L values.

Model	β	$\mathbb{D}_{enc}$	$N_{\Gamma^{\text{enc}}}$	$\mathbb{D}_{dec}$	$N_{\Gamma^{\text{dec}}}$
ours-2	0.3	2.01(0.07)	2.00(0.00)	0.89(0.10)	2.00(0.00)
	0.4	0.33(0.15)	1.60(0.55)	0.94(0.03)	2.00(0.00)
ours-4	0.3	1.30(0.09)	3.00(0.00)	0.78(0.10)	3.00(0.00)
	0.4	1.46(0.33)	3.00(0.00)	0.84(0.10)	3.00(0.00)
ours-8	0.3	1.36(0.13)	3.40(0.89)	0.62(0.17)	3.20(0.45)
	0.4	1.44(0.79)	3.40(0.55)	0.80(0.11)	3.00(0.00)
ours-16	0.3	0.60(0.31)	3.60(0.55)	0.38(0.20)	2.80(0.45)
	0.4	0.65(0.16)	3.40(0.55)	0.50(0.28)	3.00(0.71)

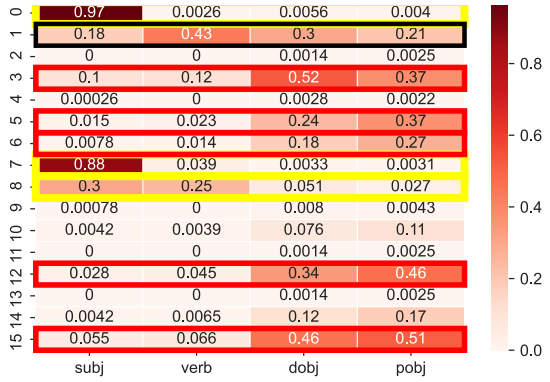


Figure B.11: Encoder influence heatmap for ADVAE with 16 latent variables on SNLI (Γ^{enc}). Squares with similar colors highlight groups of latent variables that relate to the same syntactic role.

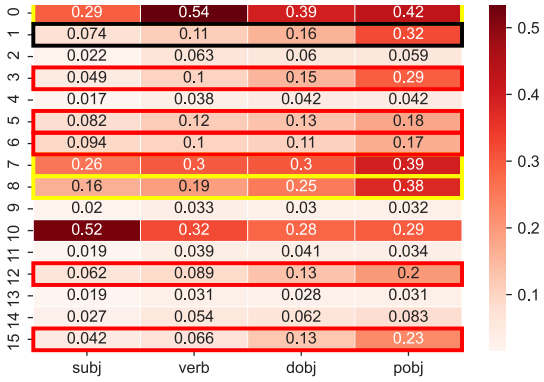


Figure B.12: Decoder influence heatmap for ADVAE with 16 latent variables on SNLI (Γ^{dec}). Squares with similar colors highlight groups of latent variables that relate to the same syntactic role.

Appendix C

Unsupervised Disentanglement of Syntax and Semantics

In this Appendix, we provide supplemental materials related to our work on unsupervised disentanglement of syntax and semantics. Appendix C.1 displays results on the development set of ParaNMT to confirm our findings, and to enable future works to compare to our work without experimenting on the test set. The hyper-parameters used for our model as well as the strategy to find them are explicited in Appendix C.2.

C.1 Results on the development set

We display, here, the scores on the development set. The encoder scores concerning the specialization of latent variables are in Table C.1, while the transfer scores are in Table C.2 for semantics, and Table C.3 for syntax. The values on the development set concerning the comparison of QKVAE with VGVAE trained on various amounts of data is in Figure C.1.

C.2 Hyper-parameters

Hyper-parameter values The β weight on the KL divergence is set to 0.6 for z^c and to 0.3 for z^s, and the λ threshold for the Free-Bits strategy is set to 0.05. KL annealing is performed between steps 3K and 6K for z^{sem}, and between steps 7K and 20K for z^{syn}. The model is trained using Adafactor [Shazeer and Stern, 2018], a memory-efficient version of

	$z^{sem}\uparrow$	$z^{syn}\downarrow$
Supervised Models		
VGVAE	99.0	16.4
SynPG	91.6	31.2
Unsupervised Models		
Optimus	89.4	-
ADVAE	41.0	40.3
QKVAE	86.7	27.0

Table C.1: The probability*100 that an embedding places a target sentence closer to its semantic source than it is to its syntactic source in the embedding space. (development set results)

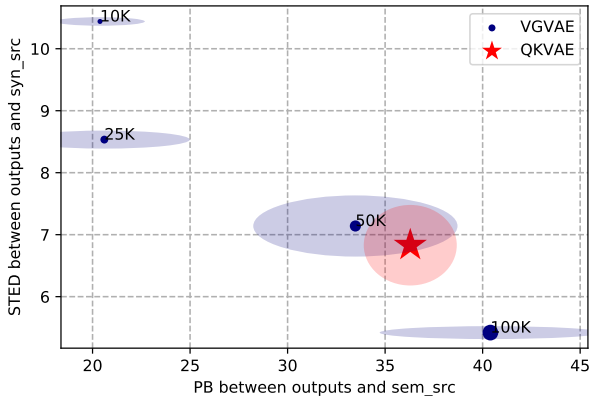


Figure C.1: Plotting STED w.r.t *syn_ref* and the PB cosine similarity w.r.t *sem_ref* for VGVAE with different amounts of labeled data and for QKVAE. Points are scaled proportionally to the amount of training data. The vertical and horizontal diameters of each ellipse are equal to the standard deviation of the associated data points and axes.

	sem_src		*syn_src*		*target*	
	M↑	*PB*↑	*M*↓	*PB*↓	*M*↑	*PB*↑
Control and Reference baselines						
sem_src	100	1.0	7.4	0.13	27.4	0.82
syn_src	7.4	0.13	100	1.0	12.0	0.16
Optimus	13.00	0.35	13.4	0.34†	10.5	0.32
Supervised Models						
VGVAE	18.3	0.58	15.2	0.17	23.0	0.57
SynPG	47.6	0.86	7.8	0.11	24.4	0.73
Unsupervised Models						
ADVAE	9.0	0.20	8.1	0.17	7.7	0.19
QKVAE	13.4	0.36	11.3	0.19	12.9	0.35

Table C.2: Semantic transfer results (development set results). The comparison scores between sentences and *syn_src* that are not significantly different from the same scores produced with regard to *sem_src* are marked with †.

	sem_src			*syn_src*			*target*		
	STED↑	*TMA2*↓	*TMA3*↓	*STED*↓	*TMA2*↑	*TMA3*↑	*STED*↓	*TMA2*↑	*TMA3*↑
Control/Ceiling baselines									
sem_src	0.0	100	100	11.9	46.4	6.8	10.9	47.0	7.3
syn_src	11.9	46.4	6.8	0.0	100	100	6.0	81.6	45.0
Optimus	9.7	58.2	20.6	9.2†	61.6†	22.6†	9.9	59.6	18.4
Supervised Models									
VGVAE	11.9	45.4	6.8	3.2	84.2	58.2	6.7	77.6	39.0
SynPG	9.3	49.4	21.4	12.2	73.0	12.2	12.2	68.6	13.0
Unsupervised Models									
ADVAE	10.1	53.4	18.6	9.8†	55.0†	17.4†	10.5	52.8	15.4
QKVAE	11.4	45.0	9.1	6.8	66.4	37.4	8.6	63.0	26.9

Table C.3: Syntactic transfer results (development set results). The comparison scores between sentences and *syn_src* that are not significantly different from the same scores produced with regard to *sem_src* are marked with †.

Adam [Kingma and Ba, 2015]. Using a batch size of 64, we train for 40 epochs, which takes about 30 hours on a single Nvidia GEForce RTX 2080 GPU. We use 4 layers for both Transformer encoders and decoders. The encoders (resp. decoders) are initialized with parameters from the 4 first layers (resp. 4 last layers) of BART encoders (resp. decoders). In total, our model uses 236M parameters.

Manual hyper-parameter search Given that the architecture for Transformer layers is fixed by BART, we mainly explored 3 parameters: number of latent variables L, number of Transformer layers, and values for β. Our first experiments have shown that setting L to 8 or 16 does not yield good results, which is probably due to the fact that a high L raises the search space for possible arrangements of values with keys, and consequently makes convergence harder. Concerning the number of layers, we observed that results with the full BART model (6 layers) have high variance over different runs. Reducing the number of layers to 4 solved this issue. In regards to β, we observed that it must be 0.6 or less for the model to produce adequate reconstructions and that it is beneficial to set it slightly lower for z^{syn} than for z^{sem} so as to absorb more syntactic information with z^{syn}.