TÉCNICO LISBOA

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Learnable Sparsity and Weak Supervision for Data-Efficient, Transparent, and Compact Neural Models

Gonçalo Migueis de Matos Afonso Correia

Supervisor : Doctor André Filipe Torres Martins
Co-Supervisor : Doctor Vlad Niculae

Thesis approved in public session to obtain the PhD Degree in Electrical and Computer Engineering

Jury final classification: Pass with Distinction and Honour

2022

TÉCNICO LISBOA

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Learnable Sparsity and Weak Supervision for Data-Efficient, Transparent, and Compact Neural Models

Gonçalo Migueis de Matos Afonso Correia

Supervisor : Doctor André Filipe Torres Martins
Co-Supervisor : Doctor Vlad Niculae

Thesis approved in public session to obtain the PhD Degree in Electrical and Computer Engineering

Jury final classification: Pass with Distinction and Honour

Jury

Chairperson : Doctor Isabel Maria Martins Cardoso, Instituto Superior Técnico, Universidade de Lisboa

Members of the Committee :

Doctor Mário Alexandre Teles de Figueiredo, Instituto Superior Técnico, Universidade de Lisboa;
Doctor Ivan Titov, School of Informatics, University of Edinburgh, UK;
Doctor André Filipe Torres Martins, Instituto Superior Técnico, Universidade de Lisboa;
Doctor Wilker Ferreira Aziz, Institute for Logic, Language and Computation, University of Amsterdam, Netherlands

Funding Institutions - European Research Council and Instituto de Telecomunicações

2022

Abstract

Neural network models have become ubiquitous in the machine learning literature. These models are compositions of differentiable building blocks that compute dense representations of the underlying data. To obtain good representations, conventional neural models require many training data points. Moreover, the representations obtained by neural models are largely uninterpretable, albeit capable of leading to high performance on many tasks. Neural models are also often overparameterized and give out representations that do not compactly represent the data. To address these issues, this thesis contributes with several solutions leveraging **sparsity** and various forms of **weak supervision**. For **data-efficiency**, we leverage transfer learning as a form of weak supervision. The proposed model can perform similarly to models trained on millions of data points on a sequence-to-sequence generation task, even though we only train it on a few thousand. For **transparency**, we propose a normalization function that can learn its sparsity. The model learns how sparse it needs to be at each layer, adapting the sparsity according to the neural component's role in the overall structure. At no cost in accuracy, sparsity helps to uncover different specializations of the neural components, aiding the interpretability of a popular neural machine translation architecture. Finally, for **compactness**, we develop a procedure to efficiently obtain deterministic gradients of discrete and structured latent variable models. The discrete nodes in these models can compactly represent implicit clusters and structures in the data. Still, their training can often be complex and prone to failure since it usually requires approximations that rely on sampling or relaxations. We propose to train these models with deterministic gradients by parameterizing discrete distributions with sparse functions, both unstructured and structured. We obtain good performance on three latent variable model applications while still achieving the practicality of the approximations mentioned above. Through these novel contributions, we challenge the conventional wisdom that neural models cannot exhibit data efficiency, transparency, or compactness.

Keywords: Machine learning, natural language processing, neural networks, sparsity, latent variable models.

Resumo

Em aprendizagem automática, os modelos baseados em redes neuronais tornaram-se omnipresentes no estado da arte. A composição destes modelos baseia-se em blocos diferenciáveis que dão origem a representações vetoriais densas dos dados subjacentes. Para obter boas representações, os métodos convencionais requerem o manuseamento de muitos dados. Para além disso, embora obtenham excelente desempenho, estes modelos não são interpretáveis e não fornecem representações dos dados de forma compacta. Para resolver estes problemas, esta tese propõe soluções que envolvem **esparsidade** e várias formas de **supervisão fraca**. Para obter **eficiência** de dados, usamos técnicas de transferência de informação como uma forma de supervisão fraca. O modelo proposto tem um desempenho semelhante a modelos treinados em milhões de dados, embora tenha sido treinado em apenas poucos milhares de exemplos. Para obter **transparência**, propomos uma função de normalização que tem a capacidade de aprender a sua própria esparsidade, ou seja, capaz de aprender a atribuir valores nulos. Esta função é diferenciável e a esparsidade pode ser por isso adaptada de acordo com os dados e de acordo com o papel que a componente neuronal do modelo em que se insere. Sem custos no desempenho, a esparsidade ajuda a descobrir especializações das componentes neuronais, ajudando a interpretabilidade de um modelo de tradução automática. Para obter **compacidade**, propomos uma maneira de obter gradientes determinísticos de forma eficiente, no treino de modelos com variáveis latentes discretas ou estruturadas. Estas componentes discretas têm a capacidade de desvendar grupos e estruturas inerentes aos dados, compactando por isso a informação. No entanto, treinar estes modelos pode ser complexo, pois exige aproximações através de amostragem ou relaxamentos para o espaço contínuo. Com a técnica utilizada neste estudo, obtemos gradientes determinísticos ao parametrizar as distribuições com funções esparsas, tanto estruturadas como não-estruturadas. Obtemos bom desempenho em três aplicações diferentes, alcançando, de qualquer forma, as vantagens práticas das aproximações acima mencionadas. Graças a estas novas contribuições científicas, a presente tese desafia a doutrina atual de que modelos neuronais não são capazes de exibir eficiência de dados, transparência, nem compacidade.

Palavras-chave: Aprendizagem automática, processamento de linguagem natural, redes neuronais, esparsidade, modelos com variáveis latentes.

Acknowledgments

This thesis would not have been possible without the support of many. Family and friends have supported me over the years to various degrees.

First of all, I dedicate this thesis to my mother, Maria da Conceição, and my father, Júlio António. I profusely thank them for everything they have done in my education, both academically and personally. I especially thank them for supporting me immediately after I told them that I wanted to pursue a Master's degree in Artificial Intelligence in Edinburgh, even though I had already started one in Biomedical Engineering. Through a domino effect, this has culminated in this thesis, and I hope I'm making them proud. I also want to thank my grandmother, Maria Helena, as she always cared for me as though I were her child. I'm also grateful to the rest of my family for their support and encouragement.

This thesis would have been much harder to do, had I not had endless support from Sofia. She has been a pillar in my life throughout these years, and I'm glad to have her as my partner. Many ideas have blossomed from conversations with her over dinner or on the long walks we usually take. Moreover, she has drawn some of the figures present in this thesis (the reader can infer which ones by how much better they look).

I am forever grateful to my official advisors, André and Vlad. They have been stellar, and I am fortunate to consider them not only advisors but also friends. I thank André for believing in me in the first place and inviting me to start on this journey with him. I thank Vlad for always looking out for me in numerous ways and inspiring me to be better.

I also thank the members of the jury for my thesis defense, Isabel Trancoso, Mário Figueiredo, Ivan Titov, and Wilker Aziz for their helpful feedback and interesting discussion during the defense. In particular, a special thanks belongs to Wilker for introducing me to latent and variational modeling, topics I was and am highly interested in, and for being kind and patient while explaining those concepts.

I am fortunate to have had the help and support of many friends. From school, I thank João Miguel, Ricardo B., Martim, Sérgio, Álvaro, Inês, Alice, Leonor, Miguel, Filipe, Ricardo F., Renato, and Luciano. I've known many of them since I was six years old, and I hope to have you by my side for countless more years! From IST (2012-2016), I thank Catarina, Patrícia, Mariana, Maria, Miguel, Joana, Diogo, Gonçalo, André Melo, Daiane, and João. They are incredible friends and an endless source of laughs, inspiration, and kindness. From

Edinburgh, I thank Alasdair, Antonio, Lasse, Diogo, and Miguel. We all shared great discussions and experiences over there, learned from among the best, and despite some physical distance right now, I still feel like we are close. I also thank my *sis-law* Inês, for being the little sister I never had, and my *bro-law* Francisco, who would undoubtedly find it extremely funny to be mentioned in a doctoral dissertation. A special thanks belongs to João Miguel, André Melo, and João Graça. When I was most confused and lost in my academic journey, they were the ones who persuaded me (thankfully!) to change paths and pursue a career in machine learning. This thesis exists, in a way, because of conversations with them.

Finally, I'm grateful for my colleagues at the SARDINE and Probabll labs, who I'm glad to call friends. I thank Ben, Erick, Tsveti, Marcos, Pedro, Chryssa, Taya, Nuno, Patrick, António, Bryan, and Lina. This endeavor would have been so dull had not all of them been around, and countless discussions led to many ideas present in this thesis. Additionally, even though we have never met in person, I thank Mathieu Blondel, who had the idea to learn α in α-entmax (Chapter 4), and had done initial work on top-k sparsemax (Chapter 5).

My PhD was funded by the European Research Council (ERC StG DeepSPIN 758969).

Je n'ai fait celle-ci plus longue que parce que
je n'ai pas eu le loisir de la faire plus courte.

Blaise Pascal, 1657

If I had more time, I would have written a shorter letter.

(Translated into English in 1658 and poetically post-edited over time)

Contents

List of Figures

List of Tables

Notation

a, $\boldsymbol{a}$, $\boldsymbol{A}$, and $\mathcal{A}$	a scalar, a vector, a matrix, and a set, respectively;
v_i	the ith element of vector $\boldsymbol{v}$;
w_{ij}	the element on the ith row and jth column of $\boldsymbol{W}$;
$\boldsymbol{e}_i$	the indicator vector for which every entry is zero, except the i^{th}, which is 1;
$\triangle^{K-1}$	the canonical simplex, *i.e.*, $\boldsymbol{\xi} \in \mathbb{R}^K : \sum_i \xi_i = 1, \boldsymbol{\xi} \geq \mathbf{0}\}$;
Y	a random variable;
$p_Y(y;\theta)$	the probability mass function (pmf) that specifies the probability distribution of the discrete random variable Y, evaluated at the outcome y using parameters θ;
$p(y\|x)$	the probability value of outcome y given outcome x;
$\text{Cat}(f(\cdot;\theta))$	the categorical distribution parameterized by the function $f(\cdot;\theta)$
$\text{Cat}(y;f(x;\theta))$	the probability mass value of outcome y under the categorical distribution parameterized by the function $f(x;\theta)$
$\mathbb{H}(p)$	the Shannon entropy of the pmf $p(z)$, *i.e.*, $-\sum_i p_i \log p_i$;
$\mathsf{KL}\left[p\|\|q\right]$	the Kullback-Leibler divergence of $p(z)$ from $q(z)$;
$\|z\|_0 \coloneqq \|\{t : z_t \neq 0\}\|$	the number of non-zeros of a sequence z, also known as the Hamming weight;
$\mathbb{E}_{p(z)}[f(z)]$	the expectation of a function $f : \mathcal{Z} \to \mathbb{R}$ under the pmf $p(z)$.

1

Introduction

Contents

Deep learning is a field of machine learning that uses neural networks to learn and obtain predictions and inferences from unseen data. The success of well-known deep learning models [Simonyan and Zisserman, 2015, Devlin et al., 2019, Brown et al., 2020, *inter alia*] relies on a rich parameterization accomplished through the use of numerous layers of computation. Along with vast amounts of data points, such a heavy parameterization allows these models to learn good vector representations that permit them to excel in their respective tasks. However, these models are often highly overparameterized; additionally, their interpretation is difficult due to the underlying opaqueness of neural models. Moreover, they require costly computing power, which causes environmental concerns [Strubell et al., 2019].

In natural language processing (NLP), one such model is the transformer architecture [Vaswani et al., 2017], which has quickly risen to prominence thanks to its outstanding performance, leading to improvements in the state of the art of neural machine translation [NMT; Junczys-Dowmunt et al., 2018a, Ott et al., 2018], and served as an inspiration to even bigger and more powerful general-purpose models like BERT [Devlin et al., 2019] and GPT-3 [Brown et al., 2020].

On the other hand, neural latent variable models are powerful and expressive tools for finding patterns in high-dimensional data, such as images or text [Kim et al., 2018, Kingma and Welling, 2014, Rezende et al., 2014]. These models have powerful structural biases that guide the model's training; of particular interest are *discrete* latent variables, which can recover discrete encodings of hidden aspects of the data, leading to compact representations [Kingma et al., 2014] and, in some cases, superior explanatory power [Titov and McDonald, 2008, Bastings et al., 2019]. However, more often than not, training models with discrete latent nodes is a difficult task due to the need to rely on high-variance methods [*e.g.*, REINFORCE; Williams, 1992] or relaxations into the continuous space [*e.g.*, Gumbel-Softmax; Jang et al., 2017, Maddison et al., 2017].

In this thesis, we will address the issues mentioned above of overuse of data points, opaqueness, and the difficulties in training compact versions of neural models. To obtain

the solutions presented, we will rely on forms of **weak supervision** and, most importantly, on **sparse** projections onto probability spaces.

When this project began in 2018, the use of large pre-trained language models was very much in an infant state. The pre-trained model ELMo [Peters et al., 2018] had just been released, closely followed by BERT [Devlin et al., 2019], and practitioners were just starting to use these models in their research to improve the state-of-the-art of tasks in NLP. The transformer arquitecture [Vaswani et al., 2017] was also a promising model that had just started to pick up much steam and to replace the recurrent neural network (RNN) sequence-to-sequence models that were prominent in the NLP literature at the time [Bahdanau et al., 2015]. During the course of this project, there has been remarkable progress in neural networks and NLP research; for example, pre-trained language model literature has evolved from proposing simply encoders that provide rich contextual representations [*e.g.*, ELMo and BERT; Peters et al., 2018, Devlin et al., 2019], to also provide decoders that deliver extremely realistic text [*e.g.*, GPT-3; Brown et al., 2020], and encoder-decoder models that allow for any task to be posed as a natural language prompt [*e.g.*, T5 and BART; Raffel et al., 2020, Lewis et al., 2020].

The approaches found in the present thesis are, in part, a product of the aforementioned research efforts. As we will see in the following chapters, we introduced a way to use large pre-trained encoders in generation tasks before large pre-trained decoder and encoder-decoder models were developed; we tackled the opaqueness of the, at the time, recently proposed and barely understood transformer architecture; and pioneered a new paradigm in discrete latent variable model training.

We also note that this thesis has strong roots in a line of literature that has been proposing new methods to induce sparsity within the computational graphs of neural networks. Before this thesis, the foundations had been laid by the study of sparse normalization functions [Martins and Astudillo, 2016, Niculae and Blondel, 2017, Peters et al., 2019] and their applications [Maruf et al., 2019, Malaviya et al., 2018], and also their counterparts and uses in structured prediction [Niculae et al., 2018a,b]. During this

project, we have extended this line of work by tackling some of the abovementioned issues with our sparsity-induced solutions. We have also created new foundations for others to extend upon our work and create their own sparse approaches to new and challenging problems [Treviso et al., 2022, Farinhas et al., 2022].

1.1 Contributions and Thesis Statement

We will now summarize the contributions of this thesis, which will address the open questions left to answer in the previous section.

- **We use weak supervision by leveraging transfer learning for data-efficient sequence-to-sequence models.** We show how to leverage a pre-trained encoder to perform a sequence-to-sequence task on a tiny dataset. We explore different avenues of parameter sharing and initialization to make this possible.

- **We propose a new deep model with attention mechanisms that can dynamically adapt to be denser or sparser as needed.** We change the transformer architecture such that each attention head, the main building block of transformers, can dynamically change its sparsity during training. This way, each attention head can accommodate its sparsity to its role in the overall model. To achieve that, we derive the gradient of α in α-entmax [Peters et al., 2019], a function akin to softmax, in which the sparsity of the probability vector is controlled by a parameter α.

- **We conduct an extensive analysis on the increased transparency of transformer models that use α-entmax as its normalization function in the attention mechanism.** Besides studying the distribution of sparsity and respective α values throughout all attention heads in the transformer, we also identify examples of sharper attention head behavior than what was found in previous work, along with the disentanglement of newfound behaviors, thanks to our proposed sparsity.

- **We propose a novel training method for discrete latent variable models that**

uses sparsity to compute an exact gradient. Thanks to this method, we can train discrete latent variable models through marginalization of the latent variable efficiently thanks to parameterizing the probability mass function with a sparse mapping: the sparsemax [Martins and Astudillo, 2016]. We test this method on a semi-supervised latent variable model and an emergent communication task. In both cases, our approach surpasses the performance of standard practices involving Monte Carlo estimation or continuous relaxations.

- **We propose two novel approaches to train structured latent variable models.** Similarly to the method for unstructured discrete latent variable models, we marginalize the structured space by using sparse mappings: SparseMAP [Niculae et al., 2018a] and the novel top-k sparsemax. In the latter, the gradient can be exact in certain conditions; in the former, the support is small from the start, making it a very competitive approach against sampling methods. We test our strategies on a bit-vector variational auto-encoder and find that they are competitive with standard approaches to training these models.

- **We provide open-source code for each of the methods we have proposed.** The respective repositories can be found in each of the chapters.

Thesis Statement. The primary claim of this thesis is that neural models do have the capability of being data-efficient, transparent, and compact: one only needs to look through a different lens that is capable of leveraging weak supervision, sparsity, and latent representations. We find that a *vanilla* application of neural models to the problem at hand is not sufficient for achieving these properties: for **data-efficiency**, we find that we need to allow parameter sharing, careful initialization, and powerful transfer learning capabilities to succeed at a low-resource generation task; for **transparency**, we can allow the model to leverage sparsity and let it learn it according to its needs at training time; and for **compactness**, we can use discrete latent variable models in a better way than what was previously possible by using a deterministic but efficient gradient.

1.2 Publications

This thesis is based, in part, on the following publications:

- **A Simple and Effective Approach to Automatic Post-Editing with Transfer Learning** [Correia and Martins, 2019]. Described in Chapter 3, this paper was accepted as a poster at ACL 2019.
- **Unbabel's Submission to the WMT2019 APE Shared Task: BERT-based Encoder-Decoder for Automatic Post-Editing** [Lopes et al., 2019]. Not included as a specific chapter within this thesis, this work describes an Automatic Post-Editing model that we submitted to the APE Shared Task at WMT2019. This model won the Shared Task, obtaining state-of-the-art in APE at the time.
- **Adaptively Sparse Transformers** [Correia et al., 2019]. Described in Chapter 4, this paper was accepted for an oral presentation at EMNLP 2019.
- **Efficient Marginalization of Discrete and Structured Latent Variables via Sparsity** [Correia et al., 2020]. Described in Chapter 5, this work was accepted as a spotlight paper at NeurIPS 2020.

1.3 Roadmap

Herein, we show the outline of the remainder of this thesis.

We begin in Chapter 2 by reviewing major concepts that are essential to understand the content of this thesis, particularly key concepts in machine learning, neural network models for NLP, sparse mappings into probability spaces, and latent variable models.

In Chapter 3, we develop a sequence-to-sequence model for Automatic Post-Editing using the transformer architecture and by harnessing the transfer learning power of a pre-trained large language model.

In Chapter 4, we modify the transformer architecture to use attention modules that

allow sparse coefficients that can adaptively change their sparsity depending on their role within the model.

In Chapter 5, we propose a new training method for structured and unstructured neural latent variable models, which uses sparsity to compute the training objective of these models, instead of using Monte Carlo methods or continuous relaxations.

Finally, in Chapter 6, we summarize the contributions of the present thesis, address some of the limitations and open problems of the present work, discuss exciting future directions of further research, and the broader impact of this thesis.

2

Background

Contents

This chapter will discuss the foundations of our work and remind the reader of crucial concepts needed to understand this thesis. Foremost, we will present key machine learning concepts. Then, we will discuss neural networks in the context of NLP, particularly sequence modeling. Secondly, we will lay the foundations for using unstructured and structured sparse projections into the simplex. Finally, we will discuss critical concepts regarding latent variable models, particularly with discrete (unstructured and structured) latent assignments.

2.1 Machine Learning

We begin by presenting the main ingredients in the common machine learning pipeline:

- a dataset $\mathcal{D}$, made out of *inputs* $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ or *input-output* pairs $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$;
- a set of parameters θ, which are the parameters of the model; and
- a loss function L.

With these ingredients, we create a machine learning pipeline by taking a *training set* that is a subset of $\mathcal{D}$ and estimate the parameters θ by minimizing the loss L on that set. The resulting model with optimal $\hat{\theta}$ can then be evaluated on a held-out subset of $\mathcal{D}$, called *test set*, to assess its performance. We will often assume $\mathcal{D}$ is made out of input-output pairs, *i.e.* we'll be addressing supervised learning, unless specified otherwise.

When the model is probabilistic, this process creates a probability model that identifies the probability measure of a random experiment: it maps from available inputs in the set of possible inputs $\mathcal{X}$ to a probability distribution of a random variable Y. When the set of possible outputs $\mathcal{Y}$ is discrete, the probability distribution of Y is prescribed by a probability mass function (pmf) $p_{Y|X}(y|\boldsymbol{x};\theta)$; when the set of possible outputs $\mathcal{Y}$ is continuous, the probability distribution of Y is prescribed by a probability density function (pdf). In the context of the present thesis, we will not have applications in which $\mathcal{Y} \in \mathbb{R}$ (*i.e.*, *regression problems*) and as such, when referring to $p_{Y|X}(y|\boldsymbol{x};\theta)$, we assume that it is

a probability mass function. Furthermore, our problems will most often involve a k-way classification, and so $p_{Y|X}(y|\boldsymbol{x};\theta)$ will be specifying a *Categorical* distribution,

$$Y|X{=}\boldsymbol{x} \sim \text{Cat}(f(\boldsymbol{x};\theta)),$$

where $f(x;\theta) \in \triangle^{K-1}$ is a function such that $p_{Y|X}(y|\boldsymbol{x};\theta) = \text{Cat}(y|f(\boldsymbol{x};\theta))$.

Supervised learning. We now turn our attention to applications in which one uses a dataset of input-output pairs. In such a case, to estimate θ, we minimize the loss over the input-pair dataset,

$$\hat{\theta} = \operatorname*{argmin}_{\theta} L_{\mathcal{D}}(\theta),$$

where $L_{\mathcal{D}}(\theta)$ is the loss function: the *negative log-likelihood*

$$L_{\mathcal{D}}(\theta) = -\mathscr{L}_{\mathcal{D}}(\theta) = -\sum_{n} \log p_{Y|X}(y_n|\boldsymbol{x}_n;\theta). \tag{2.1}$$

A model with the loss function of Equation 2.1 is called a *discriminative model*, while one that models instead $p_{XY}(\boldsymbol{x}, y;\theta)$ is called a *generative model*. Typically, the loss function might also entail a regularization term (*e.g.*, the ℓ_2 norm), which is added to the negative log-likelihood

$$L_{\mathcal{D}}(\theta) = -\mathscr{L}_{\mathcal{D}}(\theta) + \mathscr{R}(\theta).$$

Sometimes, instead of the probabilistically-grounded negative log-likelihood, practitioners might use a non-probabilistic loss ℓ and still use a statistical model; for example, one might use the 0/1 loss where the loss is 1 if the argmax mode is wrong (*i.e.*, $\hat{y} \neq y$) and 0 otherwise. After training the model, we wish to make predictions $\hat{y}$ for each novel input $\boldsymbol{x}$ of the test set, and we can make such predictions by using the trained parameters $\hat{\theta}$ and selecting, for instance, the highest-scoring $y \in \mathcal{Y}$:

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}} p_{Y|X}\left(y|\boldsymbol{x};\hat{\theta}\right).$$

For example, in a binary sentiment classification problem of English sentences, $\mathcal{Y} = \{\text{negative, positive}\}$, and $\mathcal{X}$ would be the set of possible sentences in the English language.

Unsupervised learning. Similarly, there are also applications in machine learning where we use just a dataset of inputs $\mathcal{D} = \{\boldsymbol{x}\}_{n=1}^{N}$ and estimate the parameters θ under a loss to obtain an unsupervised model; that is, a model in which inferences are made without access to any labels (*i.e.*, outputs). In this case, the loss function L only depends on the inputs $\boldsymbol{x}_n$ and the parameters θ and we thus model $p_X(\boldsymbol{x}|\theta)$. In this case, the loss function is

$$L_{\mathcal{D}}(\theta) = -\sum_{n} \log p_X(\boldsymbol{x}_n|\theta) .$$

In this setting, to assess the resulting model's performance, we often compute the log-likelihood of the model under the test set

$$\mathcal{L}_{\mathcal{D}_{\text{test}}}(\hat{\theta}) = \sum_{n} \log p_X\left(\boldsymbol{x}_n|\hat{\theta}\right) ,$$

that is, we assess the likelihood of the model being able to generate the data points $\boldsymbol{x}$.

Semi-supervised and weakly-supervised learning. In a semi-supervised setting, there is typically a smaller portion of the dataset that is labeled, $\mathcal{D}_L = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N_L}$, and the remaining majority is unlabeled, $\mathcal{D}_U = \{(\boldsymbol{x}_n)\}_{n=1}^{N_U}$. The parameters are then estimated by using a combination of losses: a component that is supervised and another that is unsupervised. Taking the loss examples given above, a probabilistic semi-supervised model could have the loss function

$$L_{\mathcal{D}}(\theta) = -\sum_{\boldsymbol{x}\in\mathcal{D}_U} \log p_X(\boldsymbol{x}|\theta) - \sum_{(\boldsymbol{x},y)\in\mathcal{D}_L} \log p_{XY}(\boldsymbol{x}, y; \theta) . \tag{2.2}$$

In some applications, both models may be independent components that share some parameters; in others, they are components of the same joint distribution over $\mathcal{X} \times \mathcal{Y}$. In the context of this thesis, we will use the term *weakly-supervised* and *weak supervision* to

refer to a broader setting that aims to alleviate the need for labeled data to obtain a well-performing model. This setting not only includes semi-supervision but also includes, for example, the transferring of learned parameters of an already optimized model to a new one, also called *transfer learning*. In such a case, the estimation of parameters can first occur by minimizing the first loss component in Equation 2.2, and then those learned parameters can be reused as an initialization on the minimization of the second component. While not explored in this thesis, other forms of weak supervision include the usage of underspecified or unreliable labels, linguistic constraints, labels obtained via heuristics, among others.

2.1.1 Linear and Deep Models

Naturally, there are many different ways one can use θ to parameterize $p_{Y|X}(y|\boldsymbol{x};\theta)$. One of the simpler ways is to define a log-linear model,

$$\begin{aligned} p_{Y|X}(y|\boldsymbol{x};\theta) &= \text{Cat}(y|f(\boldsymbol{x};\theta)) \\ f(x;\theta) &= \mathsf{softmax}(\boldsymbol{s}) \\ \boldsymbol{s} &= \boldsymbol{W}_\theta^T \boldsymbol{\phi}(\boldsymbol{x}), \end{aligned} \tag{2.3}$$

where $\boldsymbol{\phi}$ is a function that maps $\boldsymbol{x}$ into a manageable space, W_θ is a matrix of learnable weights, and **softmax** is a function that maps a vector of scores $\boldsymbol{s} \in \mathbb{R}^K$ into a "vector of probabilities", that is, $\sum_{k=1}^{K} [\mathsf{softmax}(\boldsymbol{s})]_k = 1$ and $[\mathsf{softmax}(\boldsymbol{s})]_k \geq 0$.

Definition 1: softmax

The mapping function **softmax** is defined as

$$[\mathsf{softmax}(\boldsymbol{s})]_j = \frac{\exp(s_j)}{\sum_{j'} \exp(s_{j'})}. \tag{2.4}$$

In these linear models, the problem of obtaining the optimal $\boldsymbol{W}_\theta$ lies within the set

of problems that convex optimization can solve, which significantly simplifies the optimization process. While the optimization simplicity of linear models is appealing, constructing a suitable $\boldsymbol{\phi}$ may require significant effort. The creation of an effective feature function $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^d$ might demand extensive knowledge of the domain of $\mathcal{X}$ and of the task itself. The *art* of creating these functions $\boldsymbol{\phi}$ is often called **feature engineering**.

An alternative to the log-linear model is a **neural network**, which, in a way, performs feature learning automatically through the use of hidden representations. Neural networks use non-linear functions, also known as *activation functions*, along with linear functions similar to Equation 2.3 in order to build these hidden representations.

Definition 2: neural network

A neural network with parameters θ is a function composed of building blocks that comprise both linear and non-linear functions. These building blocks are called *layers*. The layers connect in a sequence, and the output of each layer is fed into the next layer. The output of the last layer is the output of the neural network

$$f(x; \theta) = \pi \left(\boldsymbol{W}_\theta^T \boldsymbol{\phi}_\theta (\boldsymbol{x}) \right) ,$$

where compositions of differentiable functions with learnable θ parameterize $\boldsymbol{\phi}_\theta$ and π is a function that projects the scores into vectors of probabilities (*e.g.*, softmax). The composition of $\boldsymbol{\phi}_\theta$ is extremely flexible and can be defined in a variety of ways, making it a powerful way to obtain meaningful representations of the input space.

Thanks to its non-linearities, neural networks effectively create intermediate abstractions of the data, which are optimized to represent the data more effectively in order to succeed at the task, bypassing the need for feature engineering.

Just as well, non-linearities and the flexible composition of functions end up impeding the usage of convex optimization, and so neural networks are often trained instead through a gradient-based optimization algorithm based on a process called *backpropaga-*

tion.

Definition 3: backpropagation

Backpropagation is a gradient-based process that locally minimizes the loss function L by iteratively updating the parameters θ. In order to be able to update θ, the only requirement is to be able to compute the gradient of L with respect to every coordinate $\theta_d \in \theta$:

$$\frac{\partial L_{\mathcal{D}}(\theta)}{\partial \theta_d}. \tag{2.5}$$

Due to the simplicity of this requirement, practitioners can build neural networks in a modular way, where differentiable building blocks of functions are combined in arbitrary computational graphs. A handful of different algorithms can be used to update θ during backpropagation; for example, Adam [Kingma and Ba, 2015].

In practice, computing Equation 2.5 for the whole dataset $\mathcal{D}$ is a costly operation, and so the gradient is usually computed through an unbiased estimate: independent and identically distributed samples of the dataset are drawn at each iteration, and Equation 2.5 is computed on that set of samples (*i.e.*, the *mini-batch*). The size of the step at which θ is updated is determined by a value usually called the learning rate, which is a hyperparameter of the model that is chosen to maximize a given metric on a held-out dataset (the *validation set* or *development set*).

2.2 Neural Networks and Natural Language

Using neural networks in NLP applications is a pivotal part of the research presented in this thesis. While there are many ways of applying neural networks to NLP, exploiting many different architectures and aimed at many different tasks, we give special attention to sequence-to-sequence models, which we will explain in the sequel.

Nevertheless, in the context of neural networks for NLP, there is a clear focus on ob-

taining vectorized representations of words, of words within their context, and of whole sentences. A common ingredient in all neural network architectures of NLP is the use of **embeddings**. After the sentence is segmented into tokens (words, segments of words, and punctuation), the first component of the neural network is an *embedding table*: a set of learnable vectors of parameters that represent each token in the vocabulary V (chosen beforehand). For each token in the sentence, the corresponding vector (*embedding*) is taken from the table, and the resulting output is the sequential concatenation of embeddings. Many different architectures can extend the computational graph afterward, which will change those representations and entangle them to contextualize the representation of each word in the context of the sentence and the task at hand.

2.2.1 Sequence-to-Sequence Models

In NLP, predicting the next word in a sentence is a common task, for example in problems such as machine translation (MT). The broader family of models that handle such tasks are sequential models. In this thesis, we mainly focus on **sequence-to-sequence models**.

Definition 4: sequence-to-sequence models

A sequence-to-sequence model (*seq2seq*) is a model that takes as input a sequence of tokens and produces as output another sequence of tokens. The model has $Y_i|Y_{\leq i}, X \sim \text{Cat}(f(y_{\leq i}, \boldsymbol{x}; \theta))$ as its underlying distribution. In turn, the pmf

$$p(y_i|y_{\leq i}, \boldsymbol{x}) = \text{Cat}(y_i|f(y_{\leq i}, \boldsymbol{x}; \theta))$$

specifies that probability distribution, where $\boldsymbol{x} = [x^1, \ldots, x^N]$ is the input sequence of words, y_i is the next word in the output sequence $y = [y^1, \ldots, y^M]$, and $y_{\leq i}$ is the history of previous words in y. When modeled with neural networks, these typically have two main building blocks: the *encoder*, which processes $\boldsymbol{x}$, and the *decoder*, that parameterizes the probability of the next token y_i in the context of $y_{\leq i}$ and $\boldsymbol{x}$.

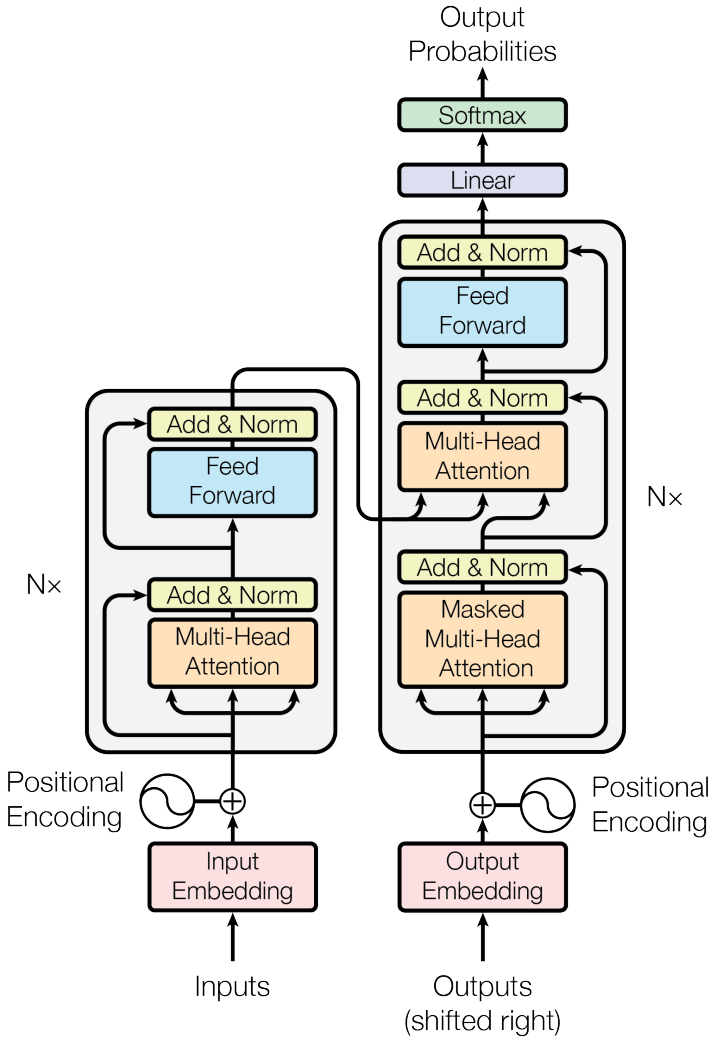


Figure 2.1: The transformer architecture [figure taken from Vaswani et al., 2017].

In the following section, we will expand on the architecture used in the sequence-to-sequence models presented in this thesis: the **transformer**.

2.2.2 Transformer

The transformer [Vaswani et al., 2017] is an architecture which maps an input sequence to an output sequence through many layers of **multi-head attention** mechanisms, yielding a dynamic, context-dependent strategy for propagating information within and across sentences. It contrasts with previous seq2seq models for neural machine translation (NMT), which usually rely either on gated recurrent operations [often LSTMs: Bahdanau et al., 2015, Luong et al., 2015] or static convolutions [Gehring et al., 2017]. A diagram of this

architecture is shown in Figure 2.1.

Attention mechanisms compute, for each query, a weighted representation of the items. The particular attention mechanism used in Vaswani et al. [2017] is called *scaled dot-product attention*.

Definition 5: attention mechanisms and scaled dot-product attention

Attention mechanisms are a differentiable building block of a neural network. For each item in the sequence, the attention mechanism computes a weighted representation of all the other items in said sequence or a given context sequence. Given the current item, it can often be interpreted as showing each item's importance in the context. A scaled dot-product attention mechanism is a particular type of attention mechanism:

$$\mathsf{Att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{\pi}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}\right)\boldsymbol{V}, \tag{2.6}$$

where, given n query contexts and m sequence items under consideration, $\boldsymbol{Q} \in \mathbb{R}^{n\times d}$ contains representations of the queries, $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{m\times d}$ are the *keys* and *values* of the items attended over, and d is the dimensionality of these representations. The $\boldsymbol{\pi}$ mapping normalizes row-wise using a mapping function like **softmax**, that is, $\boldsymbol{\pi}(\boldsymbol{Z})_{ij} = \mathsf{softmax}(\boldsymbol{z}_i)_j$. In words, the *keys* compute a relevance score between each item and query. Then, $\boldsymbol{\pi}$ normalizes these attention weights, and these will weight the *values* of each item at each query context.

However, for complex tasks, different parts of a sequence may be relevant in different ways, motivating *multi-head attention* in transformers. Multi-head attention is simply the application of Equation 2.6 in parallel H times, each with a different learned linear transformation that allows specialization,

$$\mathsf{Head}_i(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \mathsf{Att}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V). \tag{2.7}$$

In the transformer, there are three separate multi-head attention mechanisms for distinct purposes:

- **Encoder self-attention:** builds rich, layered representations of each input word by attending to the entire input sentence;
- **Context attention:** selects a representative weighted average of the encodings of the input words at each time step of the decoder;
- **Decoder self-attention:** attends over the partial output sentence fragment produced so far.

Together, these mechanisms enable the contextualized flow of information between the input sentence and the sequential decoder.

2.2.3 Large Pre-trained Language Models

A recently attractive way to obtain high-performing models in NLP is to use a pre-trained language model. These are models with many parameters, trained on millions of sentences gathered from the web, using many computational resources in the training process. Many such models are publicly released and are available for the research and industry community to use. Typically, practitioners download an already optimized model from the web and train it further by *finetuning* it, that is, taking an already trained model's parameters and optimizing them on a new dataset, usually using a lower learning rate, such that the new optimum does not veer too far from the original optimum.

Many different models have been released in the last few years, such as ELMo [Peters et al., 2018], OpenAI GPT series [Radford et al., 2018, Brown et al., 2020], BERT [Devlin et al., 2019], and T5 [Raffel et al., 2020]. Of particular interest in this thesis is BERT since it is the model used in Chapter 3.

BERT, at its core, is a transformer model, as described in §2.2.2. However, it is only made out of the encoder building block and thus does not have language modeling

capabilities in its *vanilla* form. As is the case with other similar models, BERT was trained on an enormous corpus of sentences in monolingual English and in multilingual datasets of 100+ languages. BERT's masked language modeling objective (MLM) has proven to be a powerful mechanism to learn transferrable contextual encoder representations of tokens and sentences. Many works have analyzed the BERT model and have developed variants for it. Its study has even been referred as "BERTology" [Rogers et al., 2020].

2.3 Sparsity and the Simplex

We now shift our attention to projections onto the **simplex**. Such projections can play multiple roles in a neural model. For example, as a weighting function in transformers' attention mechanisms or as a way to parameterize discrete probability distributions. The *probability simplex* $\triangle^{K-1}$ is a subset of $\mathbb{R}^K$ such that, if and only if $\boldsymbol{x} \in \triangle^{K-1}$, then $0 \leq x_k \leq 1$ for any $k \in \{1, \ldots, K\}$ and $\sum_{k=1}^{K} x_k = 1$. As aforementioned, **softmax** is one such projection that maps from $\boldsymbol{s} \in \mathbb{R}^K$ into it (Equation 2.4), but others exist, which we will explore and use in this thesis.

Note how $\mathsf{softmax}(\boldsymbol{s}) \propto \mathsf{exp}(\boldsymbol{s})$, that is, the softmax mapping (Equation 2.4) is elementwise proportional to exp; therefore, it can never assign a weight of **exactly zero**. Thus, unnecessary items are still taken into consideration to some extent. Since its output sums to one, this invariably means less weight is assigned to the relevant items, potentially harming performance and interpretability [Jain and Wallace, 2019]. This has motivated a line of research on learning networks with *sparse* mappings [Martins and Astudillo, 2016, Niculae and Blondel, 2017, Louizos et al., 2018, Shao et al., 2019]. In the following sections, we will explore projections that will be used or expanded in this thesis: sparsemax [Martins and Astudillo, 2016], α-entmax [Blondel et al., 2019, Peters et al., 2019], and SparseMAP [Niculae et al., 2018a].

2.3.1 Sparsemax

Firstly, we will introduce the **sparsemax** mapping. Like softmax, sparsemax is differentiable and has efficient forward and backward passes [Held et al., 1974, Martins and Astudillo, 2016] (*i.e.*, the transformation that happens at the input of a neural network block, and the respective gradient computation required for backpropagation, respectively), described in detail below.

Definition 6: sparsemax

The sparsemax transformation mapping [Martins and Astudillo, 2016], is given by the unique solution to the convex optimization problem

$$\mathsf{sparsemax}(\boldsymbol{s}) := \underset{\boldsymbol{\xi} \in \triangle^{K-1}}{\mathsf{argmin}} \|\boldsymbol{\xi} - \boldsymbol{s}\|_2^2 , \tag{2.8}$$

where $\boldsymbol{s} \in \mathbb{R}^K$.

Since Equation 2.8 is the Euclidean projection onto the probability simplex and solutions can hit the boundary, where some coordinates are zero-valued, sparsemax can assign **probabilities of exactly zero**; in contrast, $\mathsf{softmax}(\boldsymbol{s})$ is always dense, that is, it always maps strictly to the relative interior of the simplex where all coordinates are non-zero.

As a projection onto a polytope, the solution is likely to fall on the set's boundaries or corners. In this case, points on the border of $\triangle^{K-1}$ have one or more zero coordinates. From the optimality conditions of the sparsemax problem shown in Equation 2.8, it follows that the solution must have the form:

Lemma 1: Proof in Martins and Astudillo [2016]

$$\mathsf{sparsemax}(\boldsymbol{s}) = \mathsf{max}(\boldsymbol{s} - \tau, 0)\,, \tag{2.9}$$

where the maximum is elementwise, and τ is the unique value that ensures that the output of sparsemax sums to one.

Letting $\bar{\mathcal{Z}}$ be the set of nonzero coordinates in the solution (*i.e.*, the **support**) the normalization condition is equivalently $\tau = \frac{\sum_{z\in\bar{\mathcal{Z}}} s_z}{|\bar{\mathcal{Z}}|}$. To derive the gradient, which allows us to perform backpropagation (Equation 2.5), we can observe that small changes to $\boldsymbol{s}$ almost always have no effect on the support $\bar{\mathcal{Z}}$, and so by differentiating Equation 2.9 we obtain the backward pass for sparsemax:

$$\frac{\partial \bar{\boldsymbol{\xi}}}{\partial \bar{\boldsymbol{s}}} = \boldsymbol{I}_{|\bar{\mathcal{Z}}|} - \frac{1}{|\bar{\mathcal{Z}}|}\mathbf{1}\mathbf{1}^\top\,,$$

where $\bar{\boldsymbol{\xi}}$ and $\bar{\boldsymbol{s}}$ denote the subsets of the respective vectors indexed by the support $\bar{\mathcal{Z}}$. Outside of the support, the partial derivatives are zero (*cf.* the more general result can be found in Peters et al. [2019, Proposition 2]).

In terms of computation, τ may be found numerically using root finding algorithms on $f(\tau) = \mathsf{max}(\boldsymbol{s}-\tau, 0)-1$. Alternatively, observe that it is enough to find $\bar{\mathcal{Z}}$. By showing that sparsemax must preserve the ordering, that is, if $s_{z'} > s_z$ and $z \in \bar{\mathcal{Z}}$ then $z' \in \bar{\mathcal{Z}}$, it can be shown that $\bar{\mathcal{Z}}$ must consist of the k highest-scoring coordinates of $\boldsymbol{s}$, where k can be found by inspection after sorting $\boldsymbol{s}$. This leads to a straightforward $\mathcal{O}(K \log K)$ algorithm due to Held [Held et al., 1974, pp. 16–17]. This can be further pushed to $\mathcal{O}(K)$ using median pivoting algorithms [Condat, 2016, Peters et al., 2019]. We use a simpler implementation based on repeatedly calling top_k, doubling k until the optimal solution is found. Since solutions get sparser over time and top_k is GPU-accelerated in modern libraries [Paszke et al., 2019], this strategy is very fast in practice.

2.3.2 Entmax

We now focus on α-entmax [Blondel et al., 2019, Peters et al., 2019].

Definition 7: α-entmax

The α-entmax [Blondel et al., 2019, Peters et al., 2019] mapping is given by the unique solution to the convex optimization problem

$$\alpha\text{-entmax}(\boldsymbol{z}) := \underset{\boldsymbol{p} \in \triangle^{K-1}}{\operatorname{argmax}}\, \boldsymbol{p}^\top \boldsymbol{z} + \mathsf{H}_\alpha^{\mathrm{T}}(\boldsymbol{p}), \tag{2.10}$$

where for $\alpha \geq 1$, $\mathsf{H}_\alpha^{\mathrm{T}}$ is the Tsallis continuous family of entropies [Tsallis, 1988]:

$$\mathsf{H}_\alpha^{\mathrm{T}}(\boldsymbol{p}) := \begin{cases} \frac{1}{\alpha(\alpha-1)} \sum_j \left(p_j - p_j^\alpha\right), & \alpha \neq 1, \\ -\sum_j p_j \log p_j, & \alpha = 1. \end{cases}$$

This family contains the well-known Shannon and Gini entropies, corresponding to the cases $\alpha = 1$ and $\alpha = 2$, respectively.

Equation 2.10 involves a convex optimization subproblem. Using the definition of $\mathsf{H}_\alpha^{\mathrm{T}}$, the optimality conditions may be used to derive its solution:

Lemma 2: Proof in Peters et al. [2019]

For any $\boldsymbol{z}$, there exists a unique $\tau^\star$ such that

$$\alpha\text{-entmax}(\boldsymbol{z}) = [(\alpha - 1)\boldsymbol{z} - \tau^\star \mathbf{1}]_+^{1/(\alpha-1)}. \tag{2.11}$$

where $[\cdot]_+$ is the positive part function, $\mathbf{1}$ denotes the vector of all ones, and $\tau^\star$, which acts like a threshold, is the Lagrange multiplier corresponding to the constraint $\sum_i p_i = 1$.

Properties of α-entmax. The appeal of α-entmax rests on the following properties. For $\alpha = 1$ (*i.e.*, when $\mathsf{H}^{\mathsf{T}}_{\alpha}$ becomes the Shannon entropy) it exactly recovers the softmax mapping. For all $\alpha > 1$ it permits sparse solutions, in stark contrast to softmax. In particular, for $\alpha = 2$, it recovers the sparsemax mapping [Martins and Astudillo, 2016], which is piecewise linear. In-between, as α increases, the mapping continuously gets sparser as its curvature changes.

To compute the value of α-entmax, one must find the threshold τ such that the *r.h.s.* in Equation 2.11 sums to one. Blondel et al. [2019] propose a general bisection algorithm. Peters et al. [2019] introduce a faster, exact algorithm for $\alpha = 1.5$, and enable using α-entmax with fixed α within a neural network by showing that the α-entmax Jacobian *w.r.t.* $\boldsymbol{z}$ for $\boldsymbol{p}^\star = \alpha\text{-}\mathsf{entmax}(\boldsymbol{z})$ is

$$\frac{\partial\, \alpha\text{-}\mathsf{entmax}(\boldsymbol{z})}{\partial \boldsymbol{z}} = \mathsf{diag}(\boldsymbol{s}) - \frac{1}{\sum_j s_j}\boldsymbol{s}\boldsymbol{s}^\top, \text{ where } \quad s_i = \begin{cases} (p_i^\star)^{2-\alpha}, & p_i^\star > 0, \\ 0, & p_i^\star = 0. \end{cases}$$

Our work in §4.3 furthers the study of α-entmax by providing a derivation of the Jacobian ***w.r.t.* the parameter α**, thereby allowing the shape and sparsity of the mapping to be learned automatically. This is particularly appealing in the context of multi-head attention mechanisms, where we shall show in §4.5.1 that different heads tend to learn different sparsity behaviors.

Connections to softmax and sparsemax. The solution of sparsemax can be characterized through Equation 2.9 and thus each coordinate of the solution is a piecewise-linear function. Visibly, this expression is recovered when setting $\alpha = 2$ in the α-entmax expression (Equation 2.11); for other values of α, the exponent induces curvature.

On the other hand, softmax can be shown to be the unique solution of the optimization problem

$$\mathsf{softmax}(\boldsymbol{z})_i = \underset{\boldsymbol{p}\in\triangle^{K-1}}{\mathsf{argmax}}\, \boldsymbol{p}^\top \boldsymbol{z} + \mathbb{H}(\boldsymbol{p}),$$

where $\mathbb{H}(\boldsymbol{p})$ is the Shannon entropy. Indeed, setting the gradient to 0 yields the condition $\log p_i = z_j - \nu_i - \tau - 1$, where τ and $\nu > 0$ are Lagrange multipliers for the simplex constraints $\sum_i p_i = 1$ and $p_i \geq 0$, respectively. Since the *l.h.s.* is only finite for $p_i > 0$, we must have $\nu_i = 0$ for all i, by complementary slackness. Thus, the solution must have the form $p_i \propto \exp(z_i)$, yielding Equation 2.4.

2.3.3 SparseMAP

Theoretically, the approaches described in §2.3.1 and §2.3.2 apply to any value of K, but many interesting applications involve a K that grows exponentially as a function of sequence length. For such cases, we turn to structured prediction methods to handle the combinatorial explosion. Of particular interest in this thesis is SparseMAP [Niculae et al., 2018a,b], a structured extension of sparsemax.

Definition 8: SparseMAP

SparseMAP [Niculae et al., 2018a,b] is given by the solution of the optimization problem

$$\mathsf{SparseMAP}(\boldsymbol{t}) := \operatorname*{argmin}_{\boldsymbol{\xi} \in \triangle^{|\mathcal{Z}|-1}} \|\boldsymbol{A}\boldsymbol{\xi} - \boldsymbol{t}\|_2^2, \tag{2.12}$$

where $\boldsymbol{A} \in \mathbb{R}^{D \times K}$ is the matrix of all possible configurations of the structured variable, and $\boldsymbol{t}$ is a compact vector of variable scores of length D. The columns of $\boldsymbol{A}$ are the configurations $\boldsymbol{a}_i \in \mathbb{R}^D$ of the structured variable, and each configuration's score is given by $s_z := \langle \boldsymbol{a}_z, \boldsymbol{t} \rangle$. More on structured prediction in §2.4.2.

SparseMAP has been used successfully to model structures such as trees and matchings, and Niculae et al. [2018a] apply an active set algorithm for evaluating it and computing gradients efficiently, requiring only a primitive for computing $\mathsf{argmax}_{z \in \mathcal{Z}} \langle \boldsymbol{a}_z, \boldsymbol{t} \rangle$, which we detail below. While the argmin in Equation 2.12 is generally not unique, Carathéodory's theorem guarantees that solutions with support size at most $D + 1$ exist, and the active set algorithm enjoys linear and finite convergence to a very sparse

optimal distribution. Crucially, Equation 2.12 has a solution $\boldsymbol{\xi}^\star$ such that the set $\bar{\mathcal{Z}} = \{z \in \mathcal{Z} | \xi_z^\star > 0\}$ grows only linearly with D, and therefore $|\bar{\mathcal{Z}}| \ll |\mathcal{Z}|$.

Algorithm 2.1 Active set algorithm for SparseMAP

Init: $\bar{\mathcal{Z}}^{(0)} = \{z^{(0)}\}$ where $z^{(0)} \in \operatorname{argmax}_{z \in \mathcal{Z}} \langle \boldsymbol{a}_z, \boldsymbol{t} \rangle$ or a random structure.
for i in $1, \ldots, N$ **do**
  Compute $\tau^{(i)}$ and $\hat{\boldsymbol{\xi}}^{(i)}$ by solving the relaxed QP (Eq. 2.13). ▷ Cholesky update.
  $\boldsymbol{\xi}^{(i)} \leftarrow (1-\gamma)\boldsymbol{\xi}^{(i-1)} + \gamma \hat{\boldsymbol{\xi}}^{(i)}$ (with γ from Eq. 2.14).
  if $\gamma < 1$ **then**
    Drop the minimizer of Eq. 2.14 from $\bar{\mathcal{Z}}^{(i)}$.
  else
    Find most violated constraint, $z^{(i)} \leftarrow \operatorname{argmin}_{z \in \mathcal{Z}} \nu_z$. ▷ Eq. 2.15, MAP oracle.
    if $\nu_{z^{(i)}} \geq 0$ **then**
      return ▷ Converged.
    else
      $\bar{\mathcal{Z}}^{(i+1)} \leftarrow \bar{\mathcal{Z}}^{(i)} \cup \{z^{(i)}\}$
    end if
  end if
end for

The active set method [Nocedal and Wright, 1999, sections 16.4 & 16.5] as applied to the SparseMAP optimization problem [Equation 2.12; Niculae et al., 2018a] is a small variation of the formulation of Nocedal and Wright for handling the equality constraint, due to Martins et al. [2015, section 6]. Assume that we could identify the *support*, or *active set* of an optimal solution $\boldsymbol{\xi}^\star$, denoted $\bar{\mathcal{Z}} := \{z \in \mathcal{Z} | \xi_z^\star > 0\}$. Then, given this set, we can find the solution to Equation 2.12 by solving the lower-dimensional equality-constrained optimization problem

$$\text{minimize } \|\bar{\boldsymbol{A}}\bar{\boldsymbol{\xi}} - \boldsymbol{t}\|^2 \quad \text{s. t.} \quad \mathbf{1}^\top \bar{\boldsymbol{\xi}} = 1\,,$$

where we denote by $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{\xi}}$ the restrictions of $\boldsymbol{A}$ and $\boldsymbol{\xi}$ to the active set of structures $\bar{\mathcal{Z}}$. The solution to this equality-constrained quadratic program (QP) satisfies the Karush-Kuhn-Tucker (KKT) optimality conditions,

$$\begin{bmatrix} \bar{\boldsymbol{A}}^\top \bar{\boldsymbol{A}} & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\xi}} \\ \tau \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{A}}^\top \boldsymbol{t} \\ 1 \end{bmatrix}. \tag{2.13}$$

However, the optimal support is not known ahead of time. The active set algorithm attempts to guess the support greedily, at each iteration either (if the solution of Equation 2.13 is not feasible for Equation 2.12) dropping a structure from $\bar{\mathcal{I}}$, or (otherwise) adding a new structure. Since the support changes one structure at a time, the design matrix in Equation 2.13 gains or loses one row and column, so we may efficiently maintain its Cholesky decomposition via rank-one updates.

We now give more details about the computation. Denote the solution of Eq. 2.13, (extended with zeroes), by $\hat{\boldsymbol{\xi}} \in \triangle^{|\mathcal{I}|-1}$. Since we might not have the optimal $\bar{\mathcal{I}}$ yet, $\hat{\boldsymbol{\xi}}$ can be infeasible (some coordinates may be negative.) To account for this, we take a partial step in its direction,

$$\boldsymbol{\xi}^{(i+1)} = (1-\gamma)\boldsymbol{\xi}^{(i)} + \gamma\hat{\boldsymbol{\xi}}^{(i+1)}$$

where to ensure feasibility, the step size is given by

$$\gamma = \min\left(1, \min_{z\in\bar{\mathcal{I}};\xi_z^{(i)}>\hat{\xi}_z} \frac{\xi_z^{(i)}}{\xi_z^{(i)} - \hat{\xi}_z}\right). \tag{2.14}$$

If, on the other hand, $\hat{\boldsymbol{\xi}}$ is feasible for Equation 2.12, (so $\gamma = 1$), we check whether we have a globally optimal solution. By construction, $\hat{\boldsymbol{\xi}}$ satisfies all KKT conditions except perhaps dual feasibility $\boldsymbol{\nu} \geq 0$, where ν_z is the dual variable (Lagrange multiplier) corresponding to the constraint $\xi_z \geq 0$. Denote $\boldsymbol{\mu}^{(i)} := \boldsymbol{A}\boldsymbol{\xi}^{(i)} = \bar{\boldsymbol{A}}\bar{\boldsymbol{\xi}}^{(i)}$. For any $z \notin \bar{\mathcal{I}}$, the corresponding dual variable must satisfy

$$\nu_z = \tau^{(i)} - \langle \boldsymbol{a}_z, \boldsymbol{t} - \boldsymbol{\mu}^{(i)} \rangle . \tag{2.15}$$

If the smallest dual variable is positive, then our current guess satisfies all optimality conditions. To find the smallest dual variable we can equivalently solve $\operatorname{argmax}_{z\in\mathcal{I}} \langle \boldsymbol{a}_z, \boldsymbol{t} - \boldsymbol{\mu}^{(i)} \rangle$, which is a maximization (MAP) oracle call. If the resulting ν_z is negative, then z is the index of the most violated constraint $\xi_z \geq 0$; it is thus a good choice of structure to add to the active set.

The full procedure is given in Algorithm 2.1. The backward pass is computed by implicit differentiation of Equation 2.13 *w.r.t.* $\boldsymbol{t}$, giving, as in Niculae et al. [2018a],

$$\frac{\partial \bar{\boldsymbol{\xi}}}{\partial \boldsymbol{t}} = \bar{\boldsymbol{A}}(\boldsymbol{S} - \boldsymbol{s}\boldsymbol{s}^\top / s), \quad \text{where} \quad \boldsymbol{S} = (\bar{\boldsymbol{A}}^\top \bar{\boldsymbol{A}})^{-1}, \boldsymbol{s} = \boldsymbol{S}\mathbf{1}, s = \mathbf{1}^\top \boldsymbol{S}\mathbf{1}\,.$$

It is possible to apply the ℓ_2 regularization term only to a subset of the rows of $\boldsymbol{A}$, as is more standard in the graphical model literature. We refer the reader to the presentation in Martins et al. [2015], Niculae et al. [2018a] for this extension.

2.4 Latent Variable Models

In Chapter 5, we will focus on making neural models more compact. To that effect, we propose a novel way to train latent variable models. Latent variable models differ from the models described in §2.1, which only have *observed variables* (X and Y). Latent variables are variables that are not observed (*i.e.*, *hidden*) but are assumed to be correlated with the observed variables. Even though the training of these models may be difficult, there are significant advantages to this approach. Particularly, these models often need fewer parameters, making them more compact. Another source of compactness is that these models have built-in *inductive bias*, that is, they make assumptions about how the variables interact, yielding better interpretability and more compact representations.

Mathematically, the possibility of a latent variable being present in the modeling of $p_X(\boldsymbol{x}|\theta)$ stems from the simple observation that

$$p_X(\boldsymbol{x}|\theta) = \sum_z p(x, z) = \sum_z \pi(z|x;\theta) p_X(\boldsymbol{x}|\theta), \tag{2.16}$$

where z is the underlying latent or hidden variable and $\pi(z|x;\theta) = p(z|x;\theta)$ is the probability of an assignment z given the current observed $\boldsymbol{x}$.

When discussing latent variable models, we assume throughout that there are observed stochastic variables $x \in \mathcal{X}$ and latent stochastic variables $z \in \mathcal{Z}$. The overall fit to

a dataset $\mathcal{D}$ is $L_{\mathcal{D}}(\theta) = \sum_{x \in \mathcal{D}} \mathcal{L}_x(\theta)$, where the loss of each observation,

$$\mathcal{L}_x(\theta) = \mathbb{E}_{\pi(z|x;\theta)}\left[\ell(x, z; \theta)\right] = \sum_{z \in \mathcal{Z}} \pi(z|x; \theta)\, \ell(x, z; \theta)\ , \tag{2.17}$$

is the expected value of a downstream loss $\ell(x, z; \theta)$[1] under a probability model $\pi(z|x; \theta)$ of the latent variable; in other words, the latent variable z is *marginalized* to compute this loss. Equation 2.17 might also include a regularizer $\mathcal{R}(\theta)$. Note that, with this loss, we are assuming that the latent variable z is discrete.[2] To model complex data, one parameterizes both the downstream loss and the pmf over latent assignments using neural networks, due to their flexibility and capacity [Kingma and Welling, 2014].

2.4.1 Discrete Latent Variables

In this thesis, we will focus on **discrete** latent variables, where $|\mathcal{Z}|$ is finite but possibly very large. One example is when $\pi(z|x; \theta)$ specifies a Categorical distribution, parameterized by a vector $\boldsymbol{\xi} \in \triangle^{|\mathcal{Z}|-1}$. To obtain $\boldsymbol{\xi}$, a neural network computes a vector of scores $\boldsymbol{s} \in \mathbb{R}^{|\mathcal{Z}|}$, one score for each assignment, which is then mapped to the probability simplex, typically via $\boldsymbol{\xi} = \mathsf{softmax}(\boldsymbol{s})$. Another example is when $\mathcal{Z}$ is a structured (combinatorial) set, such as $\mathcal{Z} = \{0, 1\}^D$. In this case, $|\mathcal{Z}|$ grows exponentially with D and it is infeasible to enumerate and score all possible assignments. For this structured case, scoring assignments involves decomposition into parts, which we describe in §2.4.2.

Training such models requires summing the contributions of all assignments of the latent variable, which involves as many as $|\mathcal{Z}|$ evaluations of the downstream loss. When $\mathcal{Z}$ is not too large, the expectation may be evaluated explicitly, and learning can proceed with exact gradient updates. If $\mathcal{Z}$ is large, and/or if ℓ is an expensive computation, evaluating the expectation becomes prohibitive. In such cases, practitioners typically turn to

[1] We swapped the probabilistically-grounded $p_X(\boldsymbol{x}|\theta)$ in Equation 2.16 for a more general $\ell(x, z; \theta)$ in order to englobate any possible loss, probabilistic or non-probabilistic, and to include a optional dependency on z.

[2] We refer the reader to Kim et al. [2018] for other possibilities, such as continuous latent variables, and more details in general.

Monte Carlo (MC) estimates of $\nabla_\theta \mathcal{L}_x(\theta)$ derived from latent assignments sampled from $\pi(z|x;\theta)$. Under an appropriate learning rate schedule, this procedure converges to a local optimum of $\mathcal{L}_x(\theta)$ as long as gradient estimates are unbiased [Robbins and Monro, 1951]. Next, we describe the two current main strategies for Monte Carlo estimation of this gradient. Later, in Chapter 5, we propose our **deterministic** alternative, based on sparsifying $\pi(z|x,\theta)$.

Monte Carlo gradient estimates. Let $\theta = (\theta_\pi, \theta_\ell)$, where θ_π is the subset of weights that π depends on, and θ_ℓ the subset of weights that ℓ depends on. Given a sample $z \sim \pi(z|x;\theta_\pi)$, an unbiased estimate of the gradient for Equation 2.17 *w.r.t.* θ_ℓ is $\nabla_{\theta_\ell}\mathcal{L}_x(\theta) \approx \nabla_{\theta_\ell}\ell(x,z;\theta_\ell)$. Unbiased estimation of $\nabla_{\theta_\pi}\mathcal{L}_x(\theta)$ is more difficult, since θ_π is involved in the sampling of z, but can be done with the Score Function Estimator [SFE; Rubinstein, 1976, Paisley et al., 2012]: $\nabla_{\theta_\pi}\mathcal{L}_x(\theta) \approx \ell(x,z;\theta_\ell)\ \nabla_{\theta_\pi}\log\pi(z|x;\theta_\pi)$, also known as REINFORCE [Williams, 1992]. The SFE is powerful and general, making no assumptions on the form of z or ℓ, requiring only a sampling oracle and a way to assess gradients of $\log\pi(z|x;\theta_\pi)$. However, it comes with the cost of high variance. Making the estimator practically useful requires variance reduction techniques such as baselines [Williams, 1992, Gu et al., 2016] and control variates [Wang et al., 2013, Tucker et al., 2017, Grathwohl et al., 2018]. Variance reduction can also be achieved with Rao-Blackwellization techniques such as sum and sample [Casella and Robert, 1996, Ranganath et al., 2014, Liu et al., 2019], which marginalizes an expectation over the top-k elements of $\pi(z|x;\theta_\pi)$ and takes a sample estimate from the complement set.

Continuous relaxations. For continuous latent variables, low-variance pathwise gradient estimators can be obtained by separating the source of stochasticity from the sampling parameters, using the so-called *reparameterization trick* [Kingma and Welling, 2014, Rezende et al., 2014]. For discrete latent variables, reparameterizations can only be obtained by introducing a step function like argmax, which has null gradients almost everywhere. Replacing the gradient of argmax with a nonzero surrogate like the identity

function, known as straight-through [Bengio et al., 2013], or with the gradient of softmax, known as *Gumbel-Softmax* [Maddison et al., 2017, Jang et al., 2017], leads to a biased estimator that can still perform well in practice. Continuous relaxations like Straight-Through and Gumbel-Softmax are only possible under a further modeling assumption that ℓ is defined continuously (thus differentiably) in a neighborhood of the indicator vector $\boldsymbol{z} = \boldsymbol{e}_z$ for every $z \in \mathcal{Z}$.

2.4.2 Structured Latent Variables

Many models of interest involve structured (or combinatorial) latent variables. In this thesis, we assume that a latent z can be represented as a *bit-vector* — *i.e.*, a vector of discrete binary variables $\boldsymbol{a}_z \in \{0, 1\}^D$. This assignment of binary variables may involve global factors and constraints (*e.g.*, tree constraints, or budget constraints on the number of active variables, *i.e.*, $\sum_i [\boldsymbol{a}_z]_i \leq B$, where B is the maximum number of variables allowed to activate at the same time). In such structured problems, $|\mathcal{Z}|$ increases exponentially with D, making an exact evaluation of all possible $\ell(x, z; \theta)$ prohibitive.

Structured prediction typically handles this combinatorial explosion by parameterizing scores for individual binary variables and interactions within the global structured configuration, yielding a compact vector of **variable scores** $\boldsymbol{t} = \boldsymbol{g}(x; \theta) \in \mathbb{R}^D$ (*e.g.*, log-potentials for binary attributes), with $D \ll |\mathcal{Z}|$. Then, the score of some global configuration $z \in \mathcal{Z}$ is $s_z := \langle \boldsymbol{a}_z, \boldsymbol{t} \rangle$. The variable scores induce a unique Gibbs distribution over structures, given by $\pi(z|x; \theta) \propto \exp(\langle \boldsymbol{a}_z, \boldsymbol{t} \rangle)$. Equivalently, defining $\boldsymbol{A} \in \mathbb{R}^{D \times |\mathcal{Z}|}$ as the matrix with columns $\boldsymbol{a}_z$ for all $z \in \mathcal{Z}$, we consider the vector of probabilities parameterized by $\mathsf{softmax}(\boldsymbol{s})$, where $\boldsymbol{s} = \boldsymbol{A}^\top \boldsymbol{t}$. (In the unstructured case, $\boldsymbol{A} = \boldsymbol{I}$.)

In practice, however, we cannot materialize the matrix $\boldsymbol{A}$ or the global score vector $\boldsymbol{s}$, let alone compute the softmax and the sum in Equation 2.17. The SFE, however, can still be used, provided that exact sampling of $z \sim \pi(z|x; \theta)$ is feasible, and efficient algorithms exist for computing the normalizing constant $\sum_{z'} \exp(\langle \boldsymbol{a}_{z'}, \boldsymbol{t} \rangle)$ [Wainwright and Jordan, 2008], needed to compute the probability of a given sample.

3

A Simple and Effective Approach to Automatic Post-Editing with Transfer Learning

Contents

In the following chapter, we will focus on this thesis' goal of paving the way for more **data-efficient** neural network models. To achieve this, we use weak supervision in the form of transfer learning.

When the work presented in this chapter was published, the use of pre-trained transformers (§2.2.3) in NLP generation tasks had not been explored as much as it has been in recent years. For instance, the work that proposed BERT [Devlin et al., 2019] had only been recently released, and NLP researchers were quickly interested in using it in a variety of tasks. Currently, other variants of pre-trained transformers have been released, such as GPT-3 [Brown et al., 2020] and T5 [Raffel et al., 2020], which can generate natural language. However, as BERT was a transformer encoder, it was not straightforward to consider it to be used as a generation model.

In this work, we presented a simple and effective solution to this problem, using the scarcity of automatic post-editing [APE; Simard et al., 2007] data as a playground to explore the transfer learning capacity of a large transformer encoder model in a sequence-to-sequence task, being one of the first works to do so.

This chapter is based on Correia and Martins [2019].

3.1 Motivation

Automatic post-editing is inspired by human post-editing, in which a human translator corrects mistakes made by a machine translation (MT) system. The goal of APE is to automatically correct the mistakes produced by a black-box MT system. APE is particularly appealing for rapidly customizing MT, avoiding training new systems from scratch. Interfaces where human translators can post-edit and improve the quality of MT sentences [Alabau et al., 2014, Federico et al., 2014, Denkowski, 2015, Hokamp, 2018] are a common data source for APE models since they provide **triplets** of *source* sentences (`src`), *machine translation* outputs (`mt`), and *human post-edits* (`pe`).

Unfortunately, human post-edits are typically scarce. Existing APE systems circum-

vent this by generating **artificial triplets** [Junczys-Dowmunt and Grundkiewicz, 2016, Negri et al., 2018]. However, this requires access to a high-quality MT system, similar to (or better than) the one used in the black-box MT itself. This spoils the motivation of APE as an alternative to large-scale MT training in the first place: the time to train MT systems to extract these artificial triplets, combined with the time to train an APE system on the resulting large dataset, may well exceed the time to train an MT system from scratch.

When the current work presented was published, the state of the art in APE used a transformer [Vaswani et al., 2017] variant with **two encoders**, for the `src` and `mt`, and **one decoder**, for `pe`. This system was named the Dual-Source Transformer [Junczys-Dowmunt and Grundkiewicz, 2018, Tebbifakhr et al., 2018]. These systems greatly improved the MT baseline when concatenating human post-edited data and artificial triplets. However, few successes were known using only the actual data created by the human post-editors.

Meanwhile, there had been many successes of **transfer learning** for NLP (§2.2.3): models such as CoVe [McCann et al., 2017], ELMo [Peters et al., 2018], OpenAI GPT [Radford et al., 2018], ULMFiT [Howard and Ruder, 2018], and BERT [Devlin et al., 2019] obtain powerful representations by training large-scale language models and using them to improve performance in many sentence-level and word-level tasks. However, a language generation task such as APE presents additional challenges.

In this chapter, we build upon the successes above and show that **transfer learning is an effective and time-efficient strategy for APE**, using a pre-trained BERT model. This is an appealing strategy in practice: while large language models like BERT are expensive to train, this step is only done once and covers many languages, reducing engineering efforts substantially. In contrast, artificial triplets for APE need to be created separately for every language pair that one wishes to train an APE system for, which ends up increasing the computational and time resources in the long run.

On the other hand, using transfer learning with only the tiny shared task dataset (23K

triplets), our proposed strategy outperforms this baseline considerably, by −4.9 TER and +7.4 BLEU in the English-German WMT 2018 APE shared task, with only 3 hours of training on a single GPU. Adding the artificial eSCAPE dataset [Negri et al., 2018] leads to a performance of 17.15 TER, a new state of the art at the time.

Our main contributions are the following:

- We combine the ability of BERT to handle **sentence pair inputs** together with its pre-trained multilingual model, to use both the `src` and `mt` in a **cross-lingual encoder**, that takes a multilingual sentence pair as input.
- We show how pre-trained BERT models can also be used and fine-tuned as the **decoder** in a language generation task.
- We make a thorough empirical evaluation and ablation study of different ways of coupling BERT models in an APE system, comparing different options of parameter sharing, initialization, and fine-tuning.

3.2 Previous Work

In their Dual-Source Transformer model, Junczys-Dowmunt and Grundkiewicz [2018] also found gains by tying together encoder parameters, and the embeddings of both encoders and decoder. Our work confirms this but shows further gains by using segment embeddings and more careful sharing and initialization strategies. Sachan and Neubig [2018] explore parameter sharing between transformer layers. However, they focus on sharing decoder parameters in a *one-to-many* multilingual MT system. In our work, we share parameters between the encoder and the decoder.

As stated in §3.4, Bérard et al. [2017] also showed improved results over the MT baseline, using exclusively the shared task data. Their system outputs edit operations that decide whether to insert, keep or delete tokens from the machine-translated sentence. Instead of relying on edit operations, our approach mitigates the small amount of data

with transfer learning through BERT.

Our work makes use of key advances in transfer learning for NLP [Peters et al., 2018, Howard and Ruder, 2018, Radford et al., 2018, Devlin et al., 2019]. Pre-training these large language models has largely improved the state of the art of the GLUE benchmark [Wang et al., 2018]. Particularly, our work uses BERT [Devlin et al., 2019] and makes use of the representations obtained not only in the encoder but also on the decoder in a generation task.

More closely related to our work, Lample and Conneau [2019] pre-trained a BERT-like language model using parallel data, which they used to initialize the encoder and decoder for supervised and unsupervised MT systems. They also used segment embeddings (along with word and position embeddings) to differentiate between a pair of sentences in different languages. However, this is only used in one of the pre-training phases of the language model (translation language modeling) and not in the downstream task. In our work, we use segment embeddings during the downstream task itself, which is a perfect fit for the APE task.

3.3 Automatic Post-Editing with BERT

3.3.1 BERT as a Cross-Lingual Encoder

Our transfer learning approach is based on the bidirectional encoder representations from transformers [BERT; Devlin et al., 2019]. This model obtains deep bidirectional representations by training a transformer [Vaswani et al., 2017] with a large-scale dataset in a masked language modeling task where the objective is to predict missing words in a sentence. In our experiments, we use the $\text{BERT}_{\text{BASE}}$ model, which is composed of N=12 self-attention layers, hidden size K=768, H=12 attention heads, and feed-forward inner layer size F=3072. In addition to the word and learned position embeddings, BERT also has **segment embeddings** to differentiate between a segment A and a segment B — this is useful for tasks such as natural language inference, which involves two sentences. In

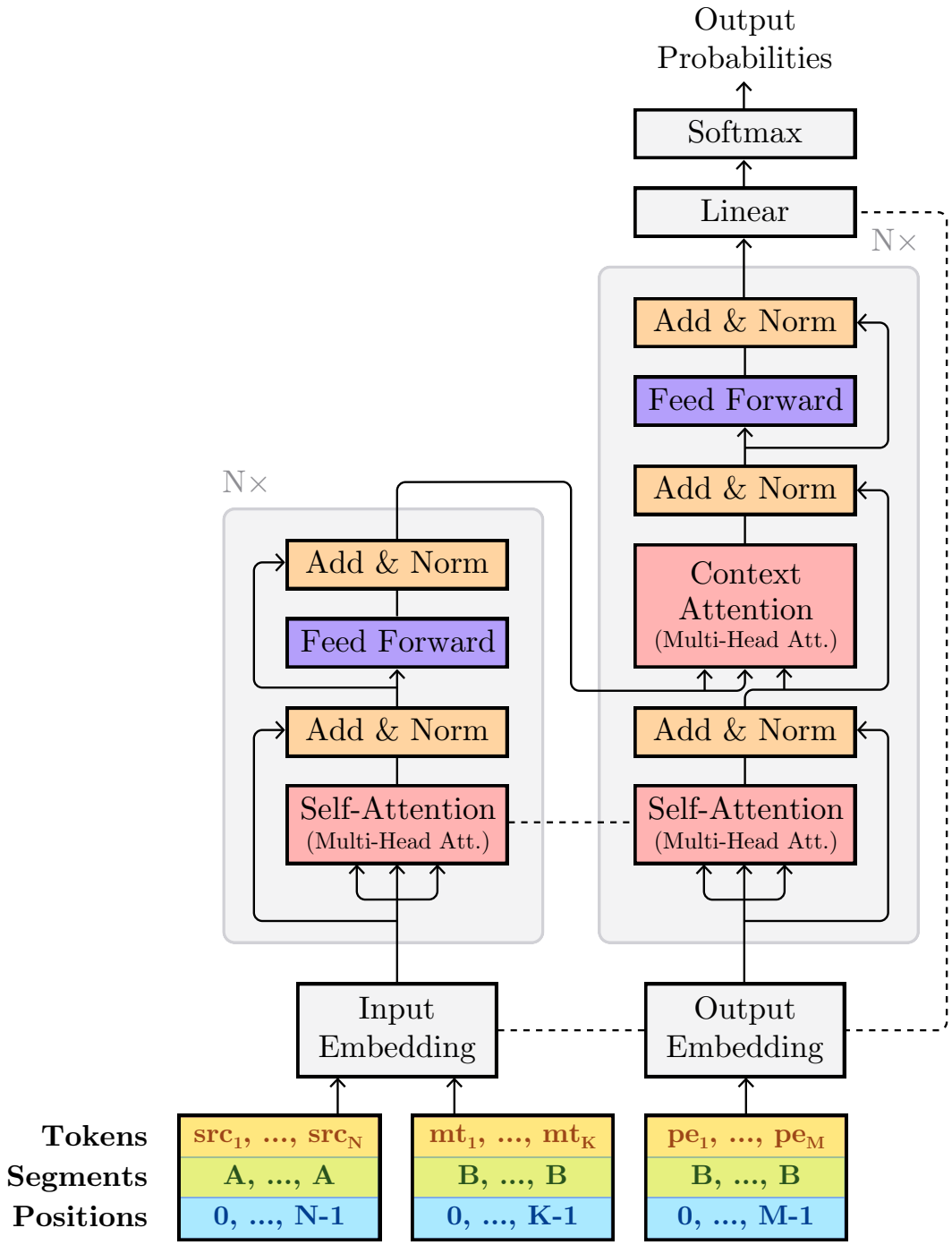


Figure 3.1: Dual-Source BERT. Dashed lines show which blocks have shared parameters in our best configuration.

the case of APE, there is also a pair of input sentences (`src`, `mt`) which are in different languages. Since one of the released BERT models was jointly pre-trained on 104 lan-

guages,[1] we use this multilingual BERT pre-trained model to encode the bilingual input pair of APE.

Therefore, the whole encoder of our APE model is the multilingual BERT: we encode both `src` and `mt` in the same encoder and use the segment embeddings to differentiate between languages (Figure 3.1). We reset positional embeddings when the `mt` starts, since it is not a continuation of `src`.

3.3.2 BERT as a Decoder

Prior work has incorporated pre-trained models as *encoders*, but not as **decoders** of sequence-to-sequence models. Doing so requires a strategy for generating fluently from the pre-trained model. Note that, when using BERT as a decoder, its bidirectionality is lost, since the model cannot look at words that have not been generated yet, and it is an open question of how to learn decoder-specific blocks (*e.g.*, context attention), which are absent in the pre-trained model.

One of our key contributions is to use BERT in the decoder by experimenting with different strategies for initializing and sharing the self and context attention layers and the position-wise feed-forward layers. We tie together the encoder and decoder embeddings weights (word, position, and segment) along with the decoder output layer (transpose of the word embedding layer). We use the same segment embedding for the target sentence (`pe`) and the second sentence in the encoder (`mt`) since they are in the same language. The full architecture is shown in Figure 3.1.

We experiment with the following strategies for coupling BERT pre-trained models in the decoder:

- **Transformer.** A transformer decoder as described in Vaswani et al. [2017] without any shared parameters, with the same dimensions as $\text{BERT}_{\text{BASE}}$ but with randomly initialized weights.

[1] https://github.com/google-research/bert/blob/eedf5716ce1268e56f0a50264a88cafad334ac61/multilingual.md

- **Pre-trained BERT.** This initializes the decoder with the pre-trained BERT model. The only component initialized randomly is the context attention (CA) layer, which is absent in BERT. Unlike in the original BERT model — which only encodes sentences — a mask in the self-attention is required to prevent the model from looking to subsequent tokens in the target sentence.

- **BERT initialized context attention.** Instead of a random initialization, we initialize the context attention layers with the weights of the corresponding BERT self-attention layers.

- **Shared self-attention.** Instead of just having the same initialization, the self-attentions (SA) in the encoder and decoder are tied during training.

- **Context attention shared with self-attention.** We take a step further and *tie* the context attention and self-attention weights — making all the attention transformation matrices (self and context) in the encoder and decoder tied.

- **Shared feed-forward.** We tie the feed-forward weights (FF) between the encoder and decoder.

3.4 Experiments

We now describe our experimental results. Our models were implemented on a fork of OpenNMT-py [Klein et al., 2017] using HuggingFace's transformers library.[2] Our model's implementation is publicly available.[3]

Datasets. We use the data from the WMT 2018 APE shared task [Chatterjee et al., 2018] (English-German SMT), which consists of 23K triplets for training, 1K for validation, and 2K for testing. In some experiments, we use the eSCAPE corpus [Negri et al., 2018], which comprises about 8M sentences; when doing so, we oversample 35x

[2] https://github.com/huggingface/transformers
[3] https://github.com/deep-spin/OpenNMT-APE

	TER↓	BLEU↑
Transformer decoder	20.33	69.31
Pre-trained BERT	20.83	69.11
with CA ← SA	18.91	71.81
and SA ↔ Encoder SA	**18.44**	**72.25**
and CA ↔ SA	18.75	71.83
and FF ↔ Encoder FF	19.04	71.53

Table 3.1: Ablation study of decoder configurations, by gradually having more shared parameters between the encoder and decoder (trained without synthetic data). ↔ denotes parameter tying and ← an initialization.

the shared task data to cover 10% of the training data. We segment words with Word-Piece [Wu et al., 2016], as used in the Multilingual BERT. At training time, we discard triplets with 200+ tokens in the combination of `src` and `mt` or 100+ tokens in `pe`. For evaluation, we use TER [Snover et al., 2006] and tokenized BLEU [Papineni et al., 2002].

Training Details. We use Adam [Kingma and Ba, 2015] with a triangular learning rate schedule that increases linearly during the first 5,000 steps until 5×10^{-5} and has a linear decay afterward. When using BERT components, we use a ℓ_2 weight decay of 0.01. We apply dropout [Srivastava et al., 2014] with $p_{\text{drop}} = 0.1$ to all layers and use label smoothing with $\epsilon = 0.1$ [Pereyra et al., 2017]. For the small data experiments, we use a batch size of 1024 tokens and save checkpoints every 1,000 steps; when using the eSCAPE corpus, we increase this to 2048 tokens and 10,000 steps. The checkpoints are created with the exponential moving average strategy of Junczys-Dowmunt et al. [2018b] with a decay of 10^{-4}. At test time, we select the model with the best TER on the development set and apply beam search with a beam size of 8 and using an average length penalty.

Model	Train Size	test 2016		test 2017		test 2018	
		TER↓	BLEU↑	TER↓	BLEU↑	TER↓	BLEU↑
MT baseline (Uncorrected)		24.76	62.11	24.48	62.49	24.24	62.99
Bérard et al. [2017]	23K	22.89	—	23.08	65.57	—	—
Junczys-Dowmunt and Grundkiewicz [2018]	5M	18.92	70.86	19.49	69.72	—	—
Junczys-Dowmunt and Grundkiewicz [2018]×4	5M	18.86	71.04	19.03	70.46	—	—
Tebbifakhr et al. [2018]	8M	—	—	—	—	18.62	71.04
Junczys-Dowmunt and Grundkiewicz [2018]	8M	17.81	72.79	18.10	71.72	—	—
Junczys-Dowmunt and Grundkiewicz [2018]×4	8M	17.34	73.43	17.47	72.84	18.00	72.52
Dual-Source Transformer†	23K	27.80	60.76	27.73	59.78	28.00	59.98
BERT Enc. + Transformer Dec. (*Ours*)	23K	20.23	68.98	21.02	67.47	20.93	67.60
BERT Enc. + BERT Dec. (*Ours*)	23K	18.88	71.61	19.03	70.66	19.34	70.41
BERT Enc. + BERT Dec. ×4 (*Ours*)	23K	**18.05**	**72.39**	**18.07**	**71.90**	**18.91**	**70.94**
BERT Enc. + BERT Dec. (*Ours*)	8M	16.91	74.29	17.26	73.42	17.71	72.74
BERT Enc. + BERT Dec. ×4 (*Ours*)	8M	**16.49**	**74.98**	**16.83**	**73.94**	**17.15**	**73.60**

Table 3.2: Results on the WMT 2016–18 APE shared task datasets. Our single models trained on the 23K dataset took only 3h20m to converge on a single Nvidia GeForce GTX 1080 GPU, while results for models trained on 8M triplets take approximately 2 days on the same GPU. Models marked with "×4" are ensembles of 4 models. Dual-Source Transformer† is a comparable re-implementation of Junczys-Dowmunt and Grundkiewicz [2018]. Best results for each training size (23K and 8M) are shown in **bold**.

Initialization and Parameter Sharing. Table 3.1 compares the different decoder strategies described in §3.3.2 on the WMT 2018 validation set. The best results were achieved by sharing the self-attention between encoder and decoder, and by initializing (but not sharing) the context attention with the same weights as the self-attention. Regarding the self-attention sharing, we hypothesize that its benefits are due to both encoder and decoder sharing a common language in their input (in the `mt` and `pe` sentence, respectively). Future work will investigate if this is still beneficial when the source and target languages are less similar. On the other hand, the initialization of the context attention with BERT's self-attention weights is essential to reap the benefits of BERT representations in the decoder — without it, using BERT decreases performance when compared to a regular transformer decoder. This might be because context attention and self-attention share the same neural block architecture (multi-head attention) and thus the context attention benefits from the pre-trained BERT's better weight initialization. No benefit was observed from sharing the feed-forward weights.

Final Results and Analysis. Finally, Table 3.2 shows our results on the WMT 2016–18 test sets. The model named *BERT Enc.* + *BERT Dec.* corresponds to the best setting found in Table 3.1, while *BERT Enc.* + *Transformer Dec.* only uses BERT in the encoder. We show results for single models and ensembles of 4 independent models.

Using the small shared task dataset only (23K triplets), our single *BERT Enc.* + *BERT Dec.* model surpasses the MT baseline by a large margin (−4.90 TER in test 2018). The only system we are aware to beat the MT baseline with only the shared task data is Bérard et al. [2017], which we also outperform (−4.05 TER in test 2017). With only about 3 GPU hours and on a much smaller dataset, our model reaches a performance that is comparable to an ensemble of the best WMT 2018 system with an artificial dataset of 5M triplets (+0.02 TER in test 2016), which is much more expensive to train. With 4× ensembling, we get competitive results with systems trained on 8M triplets.

When adding the eSCAPE corpus (8M triplets), performance surpasses the state of

the art in all test sets. By ensembling, we improve even further, achieving a final 17.15 TER score in test 2018 (−0.85 TER than the previous state of the art).

3.5 Subsequent Work

After our publication [Correia and Martins, 2019], many works have since been published exploring the use of large pre-trained language models for generation purposes (*e. g.*, [Zhang et al., 2019, Chen et al., 2020]), and our work shown in this chapter was one of the first to do so.

Most closely related to our work, and co-authored by the author of this thesis, Lopes et al. [2019] used our model on the harder English-German NMT APE subtask of WMT 2019, winning the shared task that year. To obtain this result, the transfer learning capabilities of BERT were not enough and further engineering effort was required. Particularly, a conservativeness factor was added during beam decoding to constrain the changes the APE system can make to the `mt` output. Furthermore, the authors used a data weighting method to augment the importance of data samples that have lower TER. By doing this, data samples that required less post-editing effort are assigned higher weights during the training loop. Since the NMT system does very few errors on this domain this data weighting is important for the APE model to learn to do fewer corrections to the `mt` output. However, their approach required the creation of an artificial dataset to obtain a performance that improved the MT baseline. Further investigation is required in APE to come up with better methods that obtain improved results compared to the baseline using only real post-edited data in these smaller APE datasets based on NMT outputs rather than SMT ones.

In the following years of the APE subtask, we have seen models that have been inspired by our BERT model. Particularly, Lee et al. [2020] used an XLM [Lample and Conneau, 2019] inspired model in the 2020 APE WMT subtask, instead of the multilingual BERT that our work used. Another submission to that year's subtask pre-trained

their own BERT-like model on a cross-lingual dataset, making it predict the target language during this pre-training stage [Wang et al., 2020].

On the data-efficiency side, Góis et al. [2020] performed experiments on the APE subtask, using only the real post-edited data, without resorting to expensive synthetic dataset creation. In their case, they also leveraged transfer learning using a BERT encoder, but increased the decoder's supervision to include real keystrokes data, from human post-editors. While not surpassing our approach, we consider this to also be a promising direction: increasing inductive bias in an effort to decrease larger data dependency. To address the more difficult challenge of doing APE on NMT outputs, Chollampatt et al. [2020] used our model to explore how many training instances are needed to achieve a better performance than the harder *do-nothing* NMT baseline and have concluded that, while substantial improvements for NMT require more data samples than SMT, the APE model can still improve upon the baseline with relatively few instances. To improve future APE research, they have released a more appropriately-sized dataset of real post-edited NMT data of 161K training instances based on subtitles (*SubEdits*).

The approach conducted in this chapter has also been successfully generalized to other tasks. Kodama et al. [2020] used our model for the generation of responses of a dialogue system, where, in the input, instead of `src` and a tentative `mt` output, they concatenate a question with meta-data that indicates how the system should generate a response in terms of emotion and intimacy. As in our APE case, they also achieve good performance on a low-resource dataset. Even more generally, Huang et al. [2021] identifies APE as a part of a family of generation tasks in which there are multiple inputs. Inspired by our approach, they also tackle the low-resource nature of these tasks by leveraging weak supervision through transfer learning. To mitigate catastrophic forgetting, they propose a *gradual fine-tuning* approach, in which they do an additional fine-tuning stage to adapt the pre-trained language model to a single-source sequence-to-sequence task, before finally fine-tuning on the multiple-source sequence-to-sequence task. They achieve increased performance on low-resource datasets, not only for APE but also for multi-source trans-

lation and document-level translation.

3.6 Final Remarks and Chapter Summary

In this chapter, we proposed a transfer learning approach to APE using BERT pre-trained models and careful parameter sharing. We explored various ways of coupling BERT in the decoder for language generation. We found it beneficial to initialize the context attention of the decoder with BERT's self-attention and to tie together the parameters of the self-attention layers between the encoder and decoder. Using a small dataset, our results are competitive with systems trained on a large amount of artificial data, with much faster training. By adding artificial data, we obtain a new state of the art in APE. We hope that our approach will continue to inspire future work in leveraging weak supervision through transfer learning to improve performance on generation tasks that have scarce real data.

4

Adaptively Sparse Transformers

Contents

In the previous chapter, we used transfer learning as a way to weakly supervise a transformer model on a low-resource task. While we achieved good results with this approach, the transformer architecture remains largely uninterpretable, which is unappealing if one wishes to find the reasons behind the strengths and shortcomings of the model. To this end, we introduce in this chapter an approach that uses **learnable sparsity** on each of the transformer's attention heads. In the context of this thesis, this contribution leads to neural models that are more **transparent** in their decisions. This increased transparency will prove useful in order to be able to more easily interpret the roles that each attention head ends up playing in the overall model.

This chapter is based on Correia, Niculae, and Martins [2019].

4.1 Motivation

At the heart of the transformer architecture (§2.2.2) lie *multi-head attention* mechanisms: each word is represented by multiple different weighted averages of its relevant context. As suggested in recent works on interpreting attention head roles, separate heads may learn to look for various relationships between tokens [Tang et al., 2018, Raganato and Tiedemann, 2018, Mareček and Rosa, 2018, Tenney et al., 2019, Voita et al., 2019].

The attention distribution of each head in the transformer is predicted typically using the **softmax** normalizing transformation. As a result, all context words have non-zero attention weight. Recent work on single attention architectures suggests that using sparse normalizing transformations in attention mechanisms such as sparsemax – which can yield exactly zero probabilities for irrelevant words – may improve performance and interpretability [Malaviya et al., 2018, Deng et al., 2018, Peters et al., 2019]. Qualitative analysis of attention heads [Vaswani et al., 2017, Figure 5] suggests that, depending on what phenomena they capture, heads tend to favor flatter or more peaked distributions.

Recent works have proposed sparse transformers [Child et al., 2019] and adaptive span transformers [Sukhbaatar et al., 2019]. However, the "sparsity" of those models only

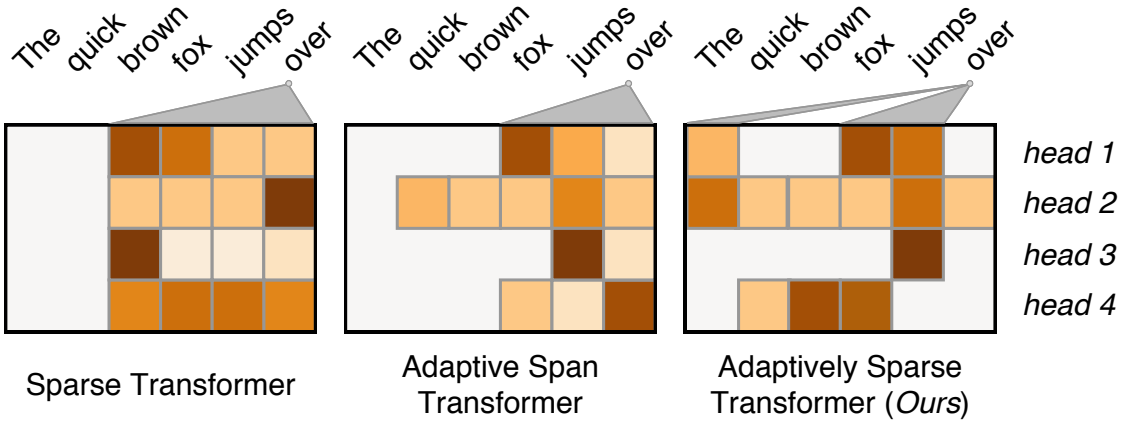


Figure 4.1: Attention distributions of different self-attention heads for the time step of the token "over", shown to compare our model to other related work. While the Sparse Transformer [Child et al., 2019] and the Adaptive Span Transformer [Sukhbaatar et al., 2019] only attend to words within a contiguous span of the past tokens, our model is not only able to obtain different and not necessarily contiguous sparsity patterns for each attention head, but is also able to tune its support over which tokens to attend adaptively.

limits the attention to a contiguous span of past tokens, while in this work we propose a **highly adaptive** transformer model that is capable of attending to a sparse set of words that are not necessarily contiguous. Figure 4.1 shows the relationship of these methods with ours.

Our contributions are the following:

- We introduce **sparse attention** into the transformer architecture, showing that it eases interpretability and leads to slight accuracy gains.
- We propose an adaptive version of sparse attention, where the shape of each attention head is **learnable** and can vary continuously and dynamically between the dense limit case of *softmax* and the sparse, piecewise-linear *sparsemax* case.[1]
- We make an extensive analysis of the added interpretability of these models, identifying both crisper examples of attention head behavior observed in previous work, as well as novel behaviors unraveled thanks to the sparsity and adaptivity of our proposed model.

[1]Code and pip package available at `https://github.com/deep-spin/entmax`.

4.2 Previous Work

Sparse attention. Prior work has developed sparse attention mechanisms, including applications to NMT [Martins and Astudillo, 2016, Malaviya et al., 2018, Niculae and Blondel, 2017, Shao et al., 2019, Maruf et al., 2019]. Peters et al. [2019] introduced the entmax function this work builds upon (§2.3.2). In their work, there is a single attention mechanism that is controlled by a fixed α. In contrast, this is the first work to allow such attention mappings to *dynamically* adapt their curvature and sparsity, by automatically adjusting the continuous α parameter. We also provide the first results using sparse attention in a transformer model.

Fixed sparsity patterns. Recent research improves the scalability of Transformer-like networks through static, fixed sparsity patterns [Child et al., 2019, Wu et al., 2019]. Our Adaptively Sparse Transformer can dynamically select a sparsity pattern that finds relevant words regardless of their position (*e.g.*, Figure 4.9). Moreover, the two strategies could be combined. In a concurrent line of research, Sukhbaatar et al. [2019] propose an adaptive attention span for transformer language models. While their work has each head learn a different contiguous span of context tokens to attend to, our work finds different sparsity patterns in the same span. Interestingly, some of their findings mirror ours — we found that attention heads in the last layers tend to be denser on average when compared to the ones in the first layers, while their work has found that lower layers tend to have a shorter attention span than higher layers.

Transformer interpretability. The original transformer paper [Vaswani et al., 2017] shows attention visualizations, from which some speculation can be made on the roles of the several attention heads. Mareček and Rosa [2018] study the syntactic abilities of the transformer self-attention, while Raganato and Tiedemann [2018] extract dependency relations from the attention weights. Tenney et al. [2019] find that the self-attentions in BERT [Devlin et al., 2019] follow a sequence of processes that resembles a classical NLP

pipeline. Regarding redundancy of heads, Voita et al. [2019] develop a method that is able to prune heads of the multi-head attention module and make an empirical study of the role that each head has in self-attention (positional, syntactic and rare words). Li et al. [2018] also aim to reduce head redundancy by adding a regularization term to the loss that maximizes head disagreement and obtain improved results. While not considering transformer attentions, Jain and Wallace [2019] show that traditional attention mechanisms do not necessarily improve interpretability since softmax attention is vulnerable to an adversarial attack leading to wildly different model predictions for the same attention weights. Sparse attention may mitigate these issues; however, our work focuses mostly on a more mechanical aspect of interpretation by analyzing head behavior, rather than on explanations for predictions.

4.3 Adaptively Sparse Transformers

We now propose a novel transformer architecture wherein we simply replace softmax with α-entmax, described in §2.3.2, in the attention heads. Concretely, we replace the row normalization $\boldsymbol{\pi}$ in Equation 2.6 by

$$\boldsymbol{\pi}(Z)_{ij} = \alpha\text{-entmax}(\boldsymbol{z}_i)_j.$$

This change leads to sparse attention weights, as long as $\alpha > 1$; in particular, $\alpha = 1.5$ is a sensible starting point [Peters et al., 2019].

Different α per head. Unlike LSTM-based seq2seq models, where α can be more easily tuned by grid search, in a transformer, there are many attention heads in multiple layers. Crucial to the power of such models, the different heads capture different linguistic phenomena, some of them isolating important words, others spreading out attention across phrases [Vaswani et al., 2017, Figure 5]. This motivates using different, adaptive α values for each attention head, such that some heads may learn to be sparser, and others

may become closer to softmax. We propose doing so by treating the α values as neural network parameters, optimized via stochastic gradients along with the other weights.

Derivatives *w.r.t.* α. In order to optimize α automatically via gradient methods, we must compute the Jacobian of the entmax output *w.r.t.* α. Since entmax is defined through an optimization problem, this is non-trivial and cannot be simply handled through automatic differentiation; it falls within the domain of *argmin differentiation*, an active research topic in optimization [Gould et al., 2016, Amos and Kolter, 2017].

One of our key contributions is the derivation of a closed-form expression for this Jacobian. The next proposition provides such an expression, enabling entmax layers with adaptive α. To the best of our knowledge, ours is the first neural network module that can automatically, and continuously vary in shape away from softmax and toward sparse mappings like sparsemax.

Proposition 1

Let $\boldsymbol{p}^\star := \alpha\text{-entmax}(\boldsymbol{z})$ be the solution of Equation 2.10,

$$\alpha\text{-entmax}(\boldsymbol{z}) := \operatorname*{argmax}_{\boldsymbol{p}\in\triangle^{K-1}} \boldsymbol{p}^\top \boldsymbol{z} + \mathsf{H}^{\mathrm{T}}_\alpha(\boldsymbol{p}).$$

Denote the distribution $\tilde{p}_i := (p_i^\star)^{2-\alpha} / \sum_j (p_j^\star)^{2-\alpha}$ and let $h_i := -p_i^\star \log p_i^\star$. The i^{th} component of the Jacobian $\boldsymbol{g} := \frac{\partial\, \alpha\text{-entmax}(\boldsymbol{z})}{\partial \alpha}$ is

$$g_i = \begin{cases} \frac{p_i^\star - \tilde{p}_i}{(\alpha-1)^2} + \frac{h_i - \tilde{p}_i \sum_j h_j}{\alpha - 1}, & \alpha > 1, \\ \frac{h_i \log p_i^\star - p_i^\star \sum_j h_j \log p_j^\star}{2}, & \alpha = 1. \end{cases} \tag{4.1}$$

The proof uses implicit function differentiation and is given in Appendix A.

Equation 4.1 provides the needed piece that was missing for training adaptively sparse transformers. In the following section, we evaluate this strategy on neural machine trans-

lation and analyze the behavior of the learned attention heads.

4.4 Experiments

We apply our adaptively sparse transformers on four MT tasks. For comparison, a natural baseline is the standard transformer architecture using the softmax transform in its multi-head attention mechanisms. We consider two other model variants in our experiments that make use of different normalizing transformations:

- **1.5-entmax:** a transformer with sparse entmax attention with fixed $\alpha = 1.5$ for all heads. This is a novel model since 1.5-entmax had only been used on RNN-based NMT models [Peters et al., 2019], but never in transformers, where attention modules are not just one single component of the model but rather an integral part of all of the model components.
- **α-entmax:** an **adaptive** transformer with sparse entmax attention with a different, learned $\alpha^t_{i,j}$ for each head.

The adaptive model has an additional scalar parameter per attention head for each of the three attention mechanisms (encoder self-attention, context attention, and decoder self-attention), *i.e.*,

$$\left\{a^t_{i,j} \in \mathbb{R} : \; i \in \{1, \dots, L\}, \; j \in \{1, \dots, H\}, \; t \in \{\texttt{enc}, \texttt{ctx}, \texttt{dec}\}\right\},$$

where L is the number of transformer layers, H is the number of attention heads in that layer, and we set $\alpha^t_{i,j} = 1 + \mathsf{sigmoid}(a^t_{i,j}) \in]1, 2[$. All or some of the α values can be tied if desired, but we keep them independent for analysis purposes.

Datasets. Our models were trained on 4 machine translation datasets of different number of training instances (sentence pairs):

- IWSLT 2017 German→English [DE→EN, Cettolo et al., 2017]: 200K.
- KFTT Japanese→English [JA→EN, Neubig, 2011]: 300K.
- WMT 2016 Romanian→English [RO→EN, Bojar et al., 2016]: 600K.
- WMT 2014 English→German [EN→DE, Bojar et al., 2014]: 4.5M.

All of these datasets were preprocessed with byte-pair encoding [BPE; Sennrich et al., 2016], using joint segmentations of 32K merge operations.

Training. We follow the dimensions of the Transformer-Base model of Vaswani et al. [2017]: The number of layers is $L = 6$ and number of heads is $H = 8$ in the encoder self-attention, the context attention, and the decoder self-attention. We use a mini-batch size of 8192 tokens and warm up the learning rate linearly until 20k steps, after which it decays according to an inverse square root schedule. All models were trained until the convergence of validation accuracy and evaluation was done at each 10K steps for RO→EN and EN→DE and at each 5K steps for DE→EN and JA→EN. The end-to-end computational overhead of our methods, when compared to standard softmax, is relatively small; in training tokens per second, the models using α-entmax and 1.5-entmax are, respectively, 75% and 90% the speed of the softmax model.

Results. We report test set tokenized BLEU [Papineni et al., 2002] results in Table 4.1. We can see that replacing softmax by entmax does not hurt performance in any of the datasets; indeed, sparse attention transformers tend to have slightly higher BLEU, but their sparsity leads to a better potential for analysis. In the next section, we make use of this potential by exploring the learned internal mechanics of the self-attention heads.

4.5 Analysis

We conduct high-level analysis of the learned attention heads of the sparse adaptive transformer model (α-entmax) trained on the 4 datasets. We then present, for the higher-

activation	DE→EN	JA→EN	RO→EN	EN→DE
softmax	29.79	21.57	32.70	26.02
1.5-entmax	29.83	**22.13**	**33.10**	25.89
α-entmax	**29.90**	21.74	32.89	**26.93**

Table 4.1: Machine translation tokenized BLEU test results on IWSLT 2017 DE→EN, KFTT JA→EN, WMT 2016 RO→EN and WMT 2014 EN→DE, respectively.

resource dataset WMT 2014 English → German, a more detailed analysis of the attention at the individual head behavior. Moreover, we make a qualitative analysis of the interpretability capabilities of our models.

4.5.1 High-Level Statistics

What kind of α values are learned? Figure 4.2 shows the learning trajectories of the α parameters of a selected subset of heads. We generally observe a tendency for the randomly-initialized α parameters to decrease initially, suggesting that softmax-like behavior may be preferable while the model is still uncertain. After around one thousand steps, some heads change direction and become sparser, perhaps as they become more confident and specialized. This shows that the initialization of α does not predetermine its sparsity level or the role the head will have throughout. In particular, head 8 in the encoder self-attention layer 2 first drops to around $\alpha = 1.3$ before becoming one of the sparsest heads, with $\alpha \approx 2$.

The overall distribution of α values at convergence can be seen in Figure 4.3. We observe that the encoder self-attention blocks learn to concentrate the α values in two modes: a very sparse one around $\alpha \rightarrow 2$, and a dense one between softmax and 1.5-entmax. However, the decoder self and context attention only learn to distribute these parameters in a single mode. We show next that this is reflected in the average density of attention weight vectors as well.

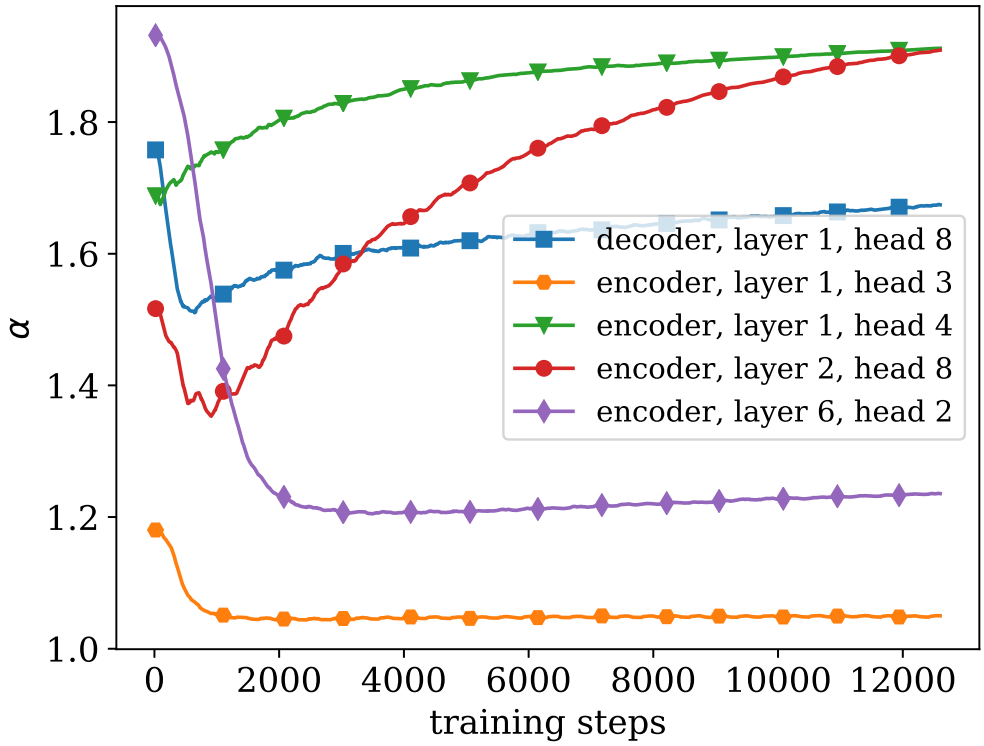
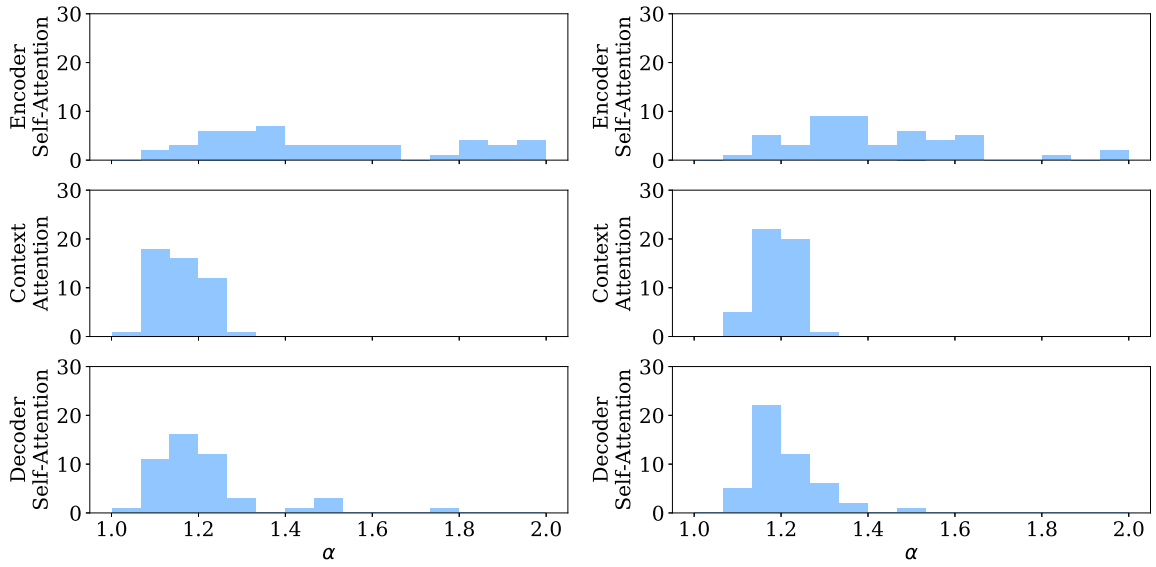


Figure 4.2: Trajectories of α values for a subset of the heads during training on EN→DE. Initialized at random, most heads become denser in the beginning, before converging. This suggests that dense attention may be more beneficial while the network is still uncertain, being replaced by sparse attention afterward.

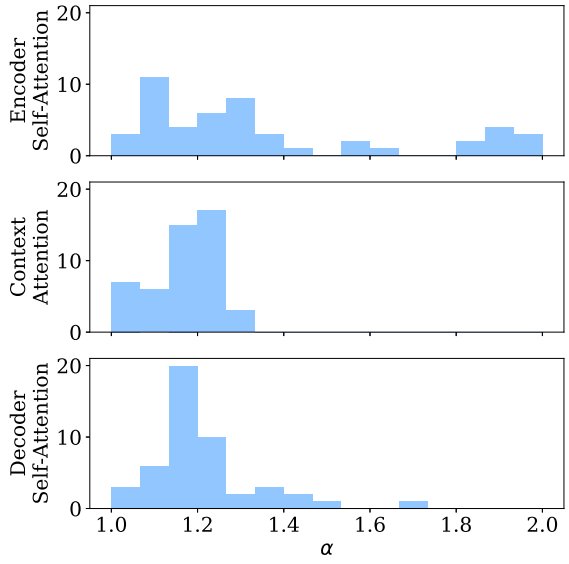
Attention weight density when translating. For any $\alpha > 1$, it would still be possible for the weight matrices in Equation 2.7 to learn re-scalings so as to make attention sparser or denser. To visualize the impact of adaptive α values, we compare the empirical attention weight density (the average number of tokens receiving non-zero attention) within each module, against sparse transformers with fixed $\alpha = 1.5$.

Figure 4.4 shows that, with fixed $\alpha = 1.5$, heads tend to be sparse and similarly distributed in all three attention modules. With learned α, there are two notable changes: (i) a prominent mode corresponding to fully dense probabilities, showing that our models learn to combine sparse and dense attention, and (ii) a distinction between the encoder self-attention — whose background distribution tends toward extreme sparsity — and the other two modules, which exhibit more uniform background distributions. This suggests

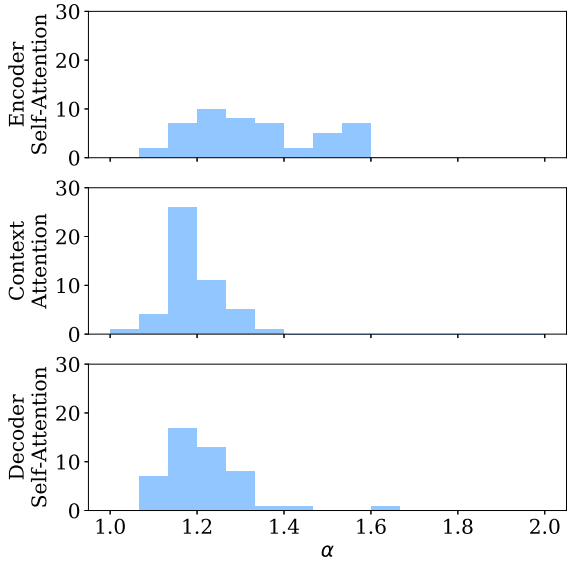


(a) WMT 2016 RO→EN.

(b) KFTT JA→EN.

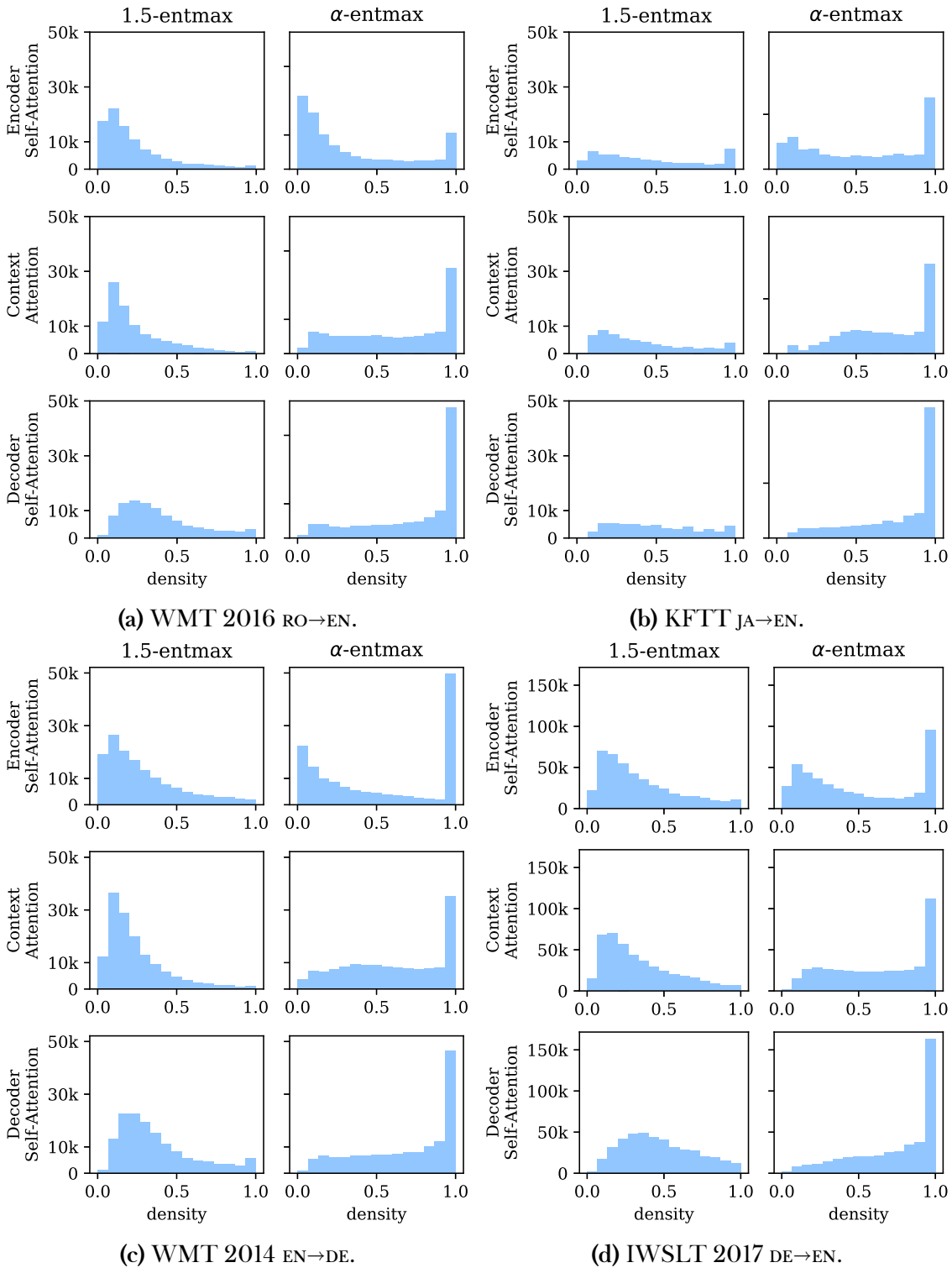


(c) WMT 2014 EN→DE.



(d) IWSLT 2017 DE→EN.

Figure 4.3: Distribution of learned α values per attention block. While the encoder self-attention has a bimodal distribution of values of α, the decoder self-attention and context attention have a single mode.



(a) WMT 2016 RO→EN.

(b) KFTT JA→EN.

(c) WMT 2014 EN→DE.

(d) IWSLT 2017 DE→EN.

Figure 4.4: Distribution of attention densities (average number of tokens receiving non-zero attention weight) for all attention heads and all validation sentences. When compared to 1.5-entmax, α-entmax distributes the sparsity more uniformly, with a clear mode at fully dense attentions, corresponding to the heads with low α. In the softmax case, this distribution would lead to a single bar with density 1.

that perhaps entirely sparse transformers are suboptimal.

The fact that the decoder seems to prefer denser attention distributions might be attributed to it being auto-regressive, only having access to past tokens and not the full sentence. We speculate that it might lose too much information if it assigned weights of zero to too many tokens in the self-attention, since there are fewer tokens to attend to in the first place.

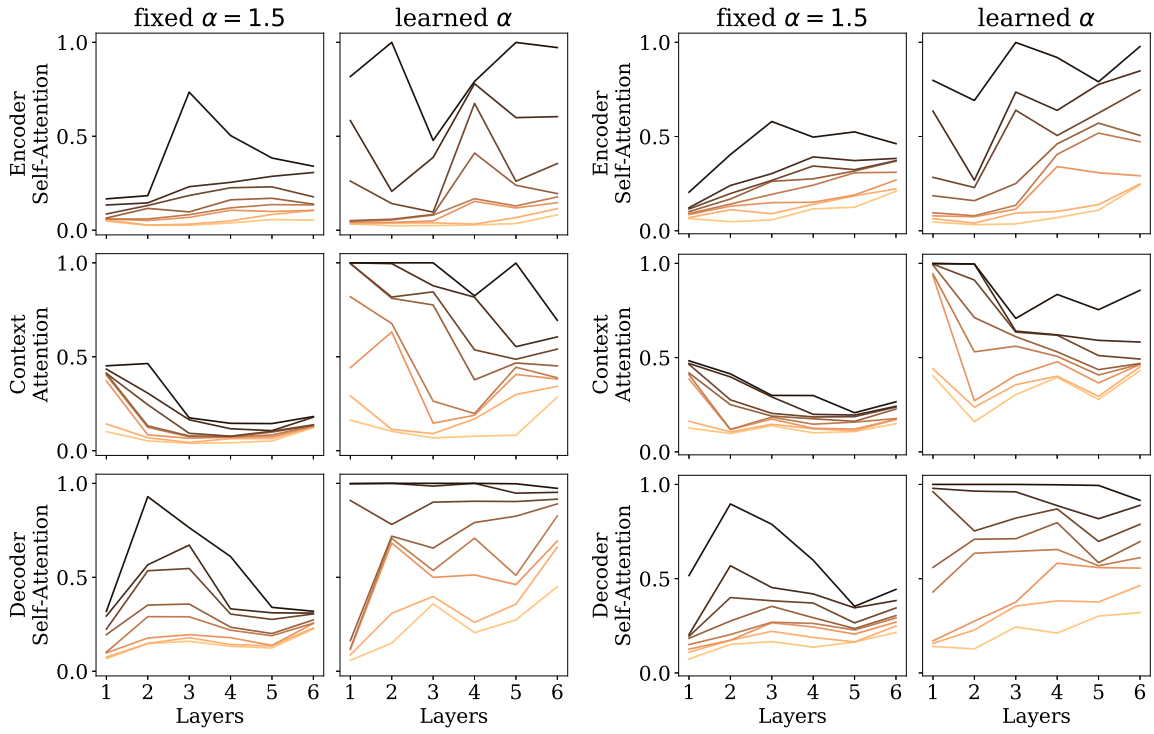
Teasing this down into separate layers, Figure 4.5 shows the average (sorted) density of each head for each layer. We observe that α-entmax is able to learn different sparsity patterns at each layer, leading to more variation in individual head behavior, to clearly-identified dense and sparse heads and overall different tendencies compared to the fixed case of $\alpha = 1.5$.

Head diversity. To measure the overall disagreement between heads, as a measure of head diversity, we use the following generalization of the Jensen-Shannon divergence:

$$\mathrm{JS} = \mathbb{H}\left(\frac{1}{H}\sum_{j=1}^{H} \boldsymbol{p}_j\right) - \frac{1}{H}\sum_{j=1}^{H} \mathbb{H}(\boldsymbol{p}_j)$$

where $\boldsymbol{p}_j$ is the vector of attention weights assigned by the head j to each word in the sequence, and $\mathbb{H}$ is the Shannon entropy, base-adjusted based on the dimension of $\boldsymbol{p}$ such that $\mathrm{JS} \leq 1$. We average this measure over the entire validation set. The higher this metric is, the more the heads are taking different roles in the model.

Figure 4.6 shows that both sparse transformer variants show more diversity than the traditional softmax one. Interestingly, diversity seems to often peak in the middle layers of the encoder self-attention and context attention, while this is not the case for the decoder self-attention.



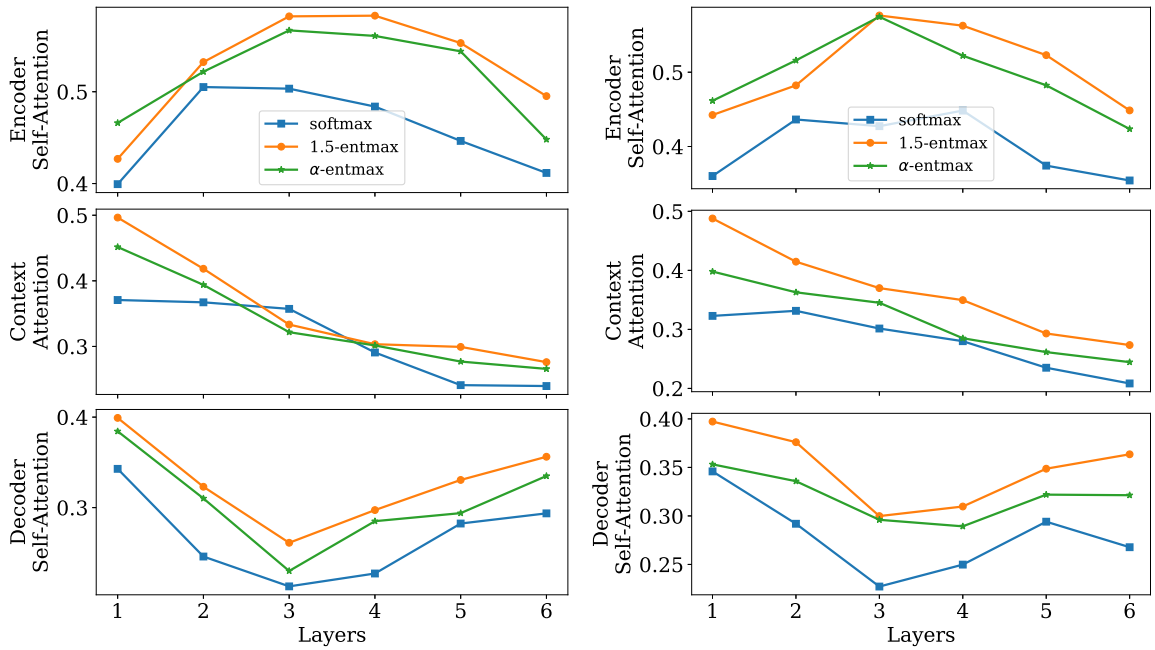
(a) WMT 2016 RO→EN.

(b) KFTT JA→EN.

(c) WMT 2014 EN→DE.

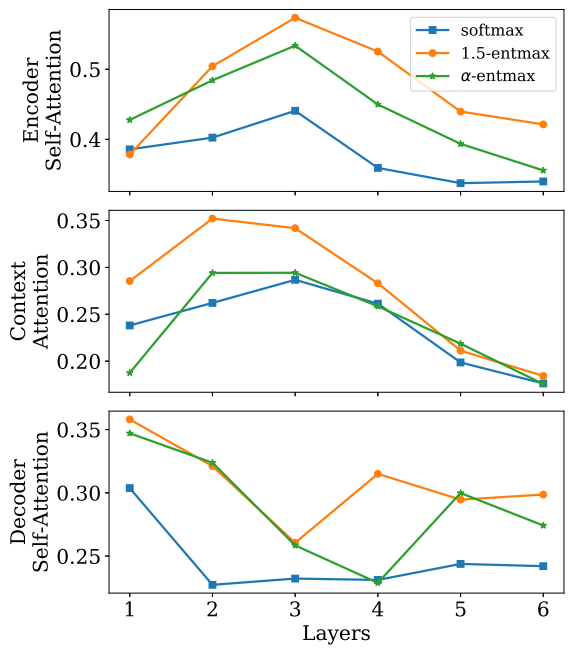
(d) IWSLT 2017 DE→EN.

Figure 4.5: Head density per layer for fixed and learned α. Each line corresponds to an attention head; lower values mean that that attention head is sparser. Learned α has higher variance.

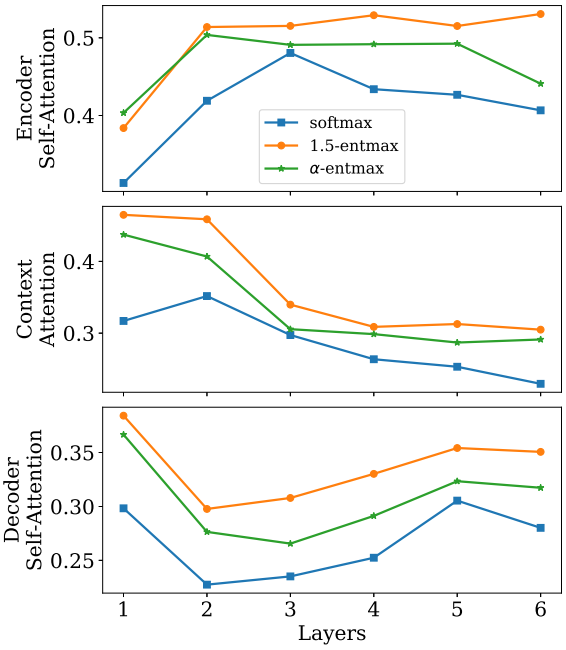


(a) WMT 2016 RO→EN.

(b) KFTT JA→EN.



(c) WMT 2014 EN→DE.



(d) IWSLT 2017 DE→EN.

Figure 4.6: Jensen-Shannon Divergence between heads at each layer. Measures the disagreement between heads: the higher the value, the more the heads are disagreeing with each other in terms of where to attend. Models using sparse entmax have more diverse attention than the softmax baseline.

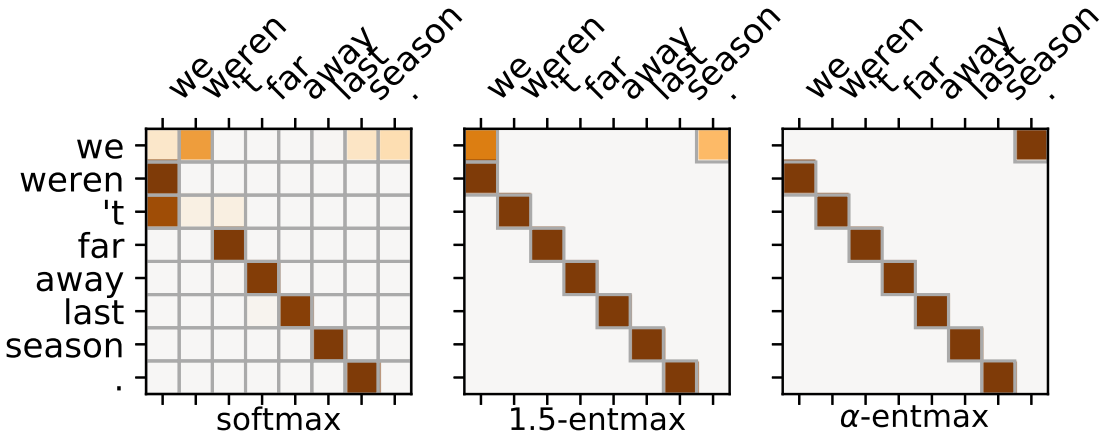


Figure 4.7: Self-attention from the most confidently previous-position head in each model. The learned parameter in the α-entmax model is $\alpha = 1.91$. Quantitatively more confident, visual inspection confirms that the adaptive head behaves more consistently.

4.5.2 Identifying Head Specializations

Previous work pointed out some specific roles played by different heads in the softmax transformer model [Voita et al., 2018, Tang et al., 2018, Voita et al., 2019]. Identifying the specialization of a head can be done by observing the type of tokens or sequences that the head often assigns most of its attention weight; this is facilitated by sparsity.

Positional heads. One particular type of head, as noted by Voita et al. [2019], is the positional head. These heads tend to focus their attention on either the previous or next token in the sequence, thus obtaining representations of the neighborhood of the current time step. In Figure 4.7, we show attention plots for such heads, found for each of the studied models. The sparsity of our models allows these heads to be more confident in their representations, by assigning the whole probability weight to a single token in the sequence. Concretely, we may measure a positional head's **confidence** as the average attention weight assigned to the previous token. The softmax model has three heads for position -1, with median confidence of 93.5%. The 1.5-entmax model also has three heads for this position, with median confidence of 94.4%. The adaptive model has four heads, with median confidences of 95.9%, the lowest-confidence head being dense with $\alpha = 1.18$, while the highest-confidence head is sparse ($\alpha = 1.91$).

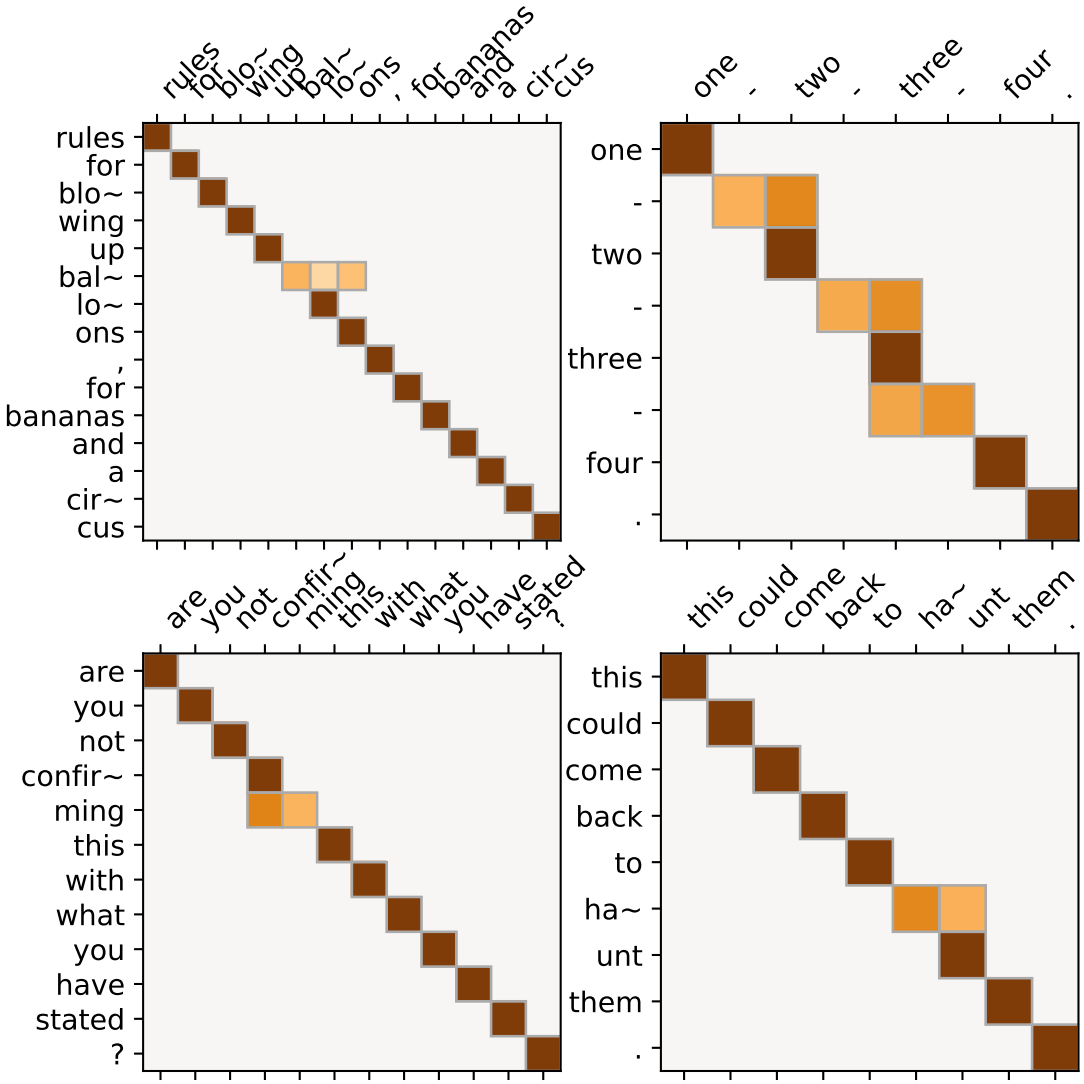


Figure 4.8: BPE-merging head ($\alpha = 1.91$) discovered in the α-entmax model. Found in the first encoder layer, this head learns to discover some subword units and combine their information, leaving most words intact. It places 99.09% of its probability mass within the same BPE cluster as the current token: more than any head in any other model.

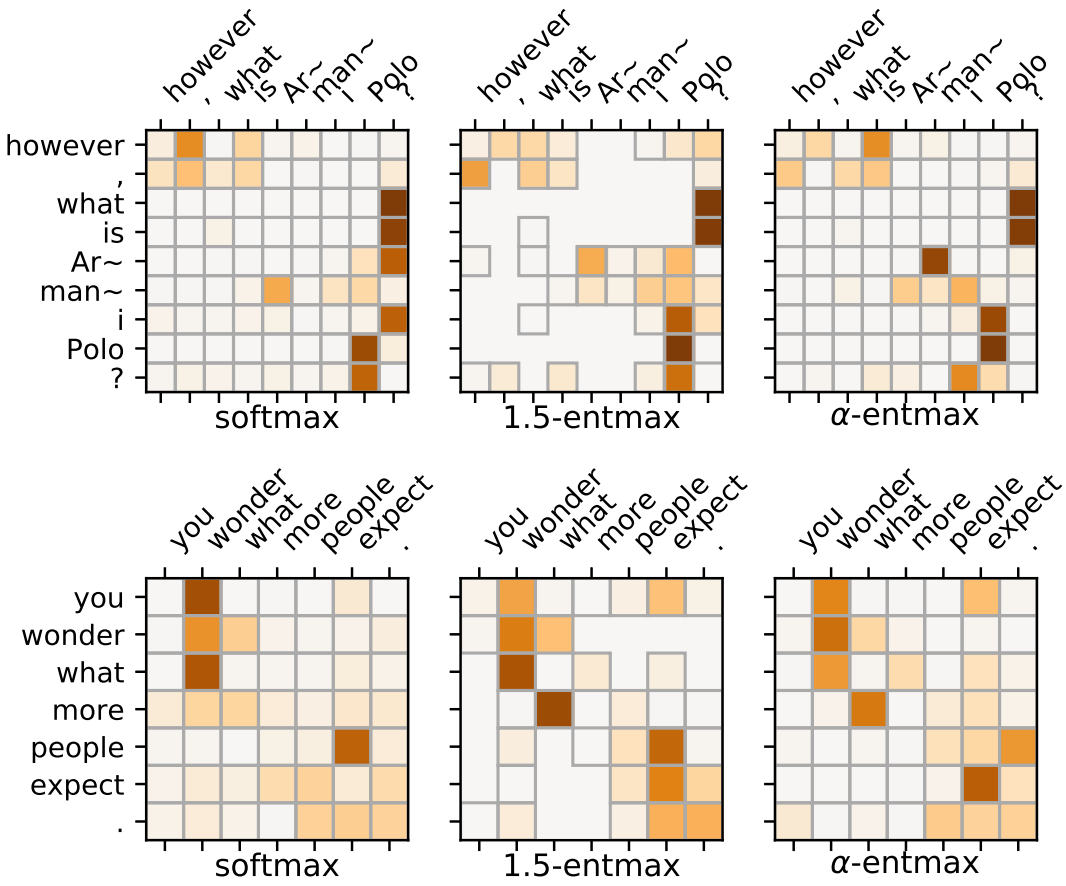


Figure 4.9: Interrogation-detecting heads in the three models. The top sentence is interrogative while the bottom one is declarative but includes the interrogative word "what". In the top example, these *interrogation heads* assign a high probability to the question mark in the time step of the interrogative word (with $\geq 97.0\%$ probability), while in the bottom example since there is no question mark, the same head does not assign a high probability to the last token in the sentence during the interrogative word time step. Surprisingly, this head prefers a low $\alpha = 1.05$, as can be seen from the dense weights. This allows the head to identify the noun phrase "Armani Polo" better.

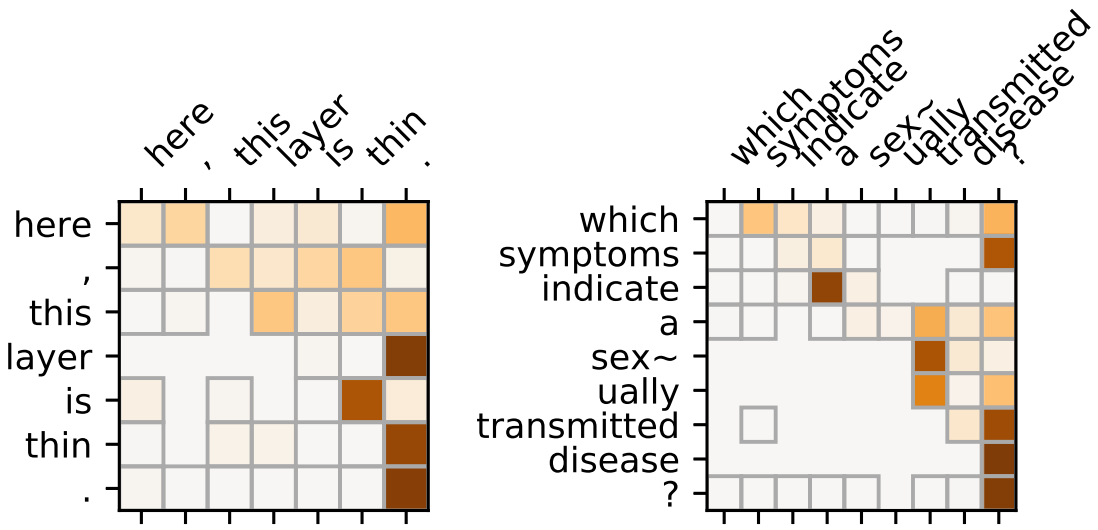


Figure 4.10: Example of two sentences of similar length where the same head ($\alpha = 1.33$) exhibits different sparsity. The longer phrase in the example on the right "a sexually transmitted disease" is handled with higher confidence, leading to more sparsity.

For position +1, the models each dedicate one head, with confidence around 95%, slightly higher for entmax. The adaptive model sets $\alpha = 1.96$ for this head.

BPE-merging head. Due to the sparsity of our models, we are able to identify other head specializations, easily identifying which heads should be further analyzed. In Figure 4.8 we show one such head where the α value is particularly high (in the encoder, layer 1, head 4 depicted in Figure 4.2). We found that this head most often looks at the current time step with high confidence, making it a positional head with offset 0. However, this head often spreads weight sparsely over 2-3 neighboring tokens, when the tokens are part of the same BPE cluster[2] or hyphenated words. As this head is in the first layer, it provides a useful service to the higher layers by combining information evenly within some BPE clusters.

For each BPE cluster or cluster of hyphenated words, we computed a score between 0 and 1 that corresponds to the maximum attention mass assigned by any token to the rest of the tokens inside the cluster in order to quantify the BPE-merging capabilities of

[2]BPE-segmented words are denoted by ~ in the Figures.

these heads.[3] There are not any attention heads in the softmax model that are able to obtain a score over 80%, while for 1.5-entmax and α-entmax there are two heads in each (83.3% and 85.6% for 1.5-entmax and 88.5% and 89.8% for α-entmax).

Interrogation head. On the other hand, in Figure 4.9 we show a head for which our adaptively sparse model chose an α close to 1, making it closer to softmax (also shown in *encoder, layer 1, head 3* depicted in Figure 4.2). We observe that this head assigns a high probability to question marks at the end of the sentence in time steps where the current token is interrogative, thus making it an interrogation-detecting head. We also observe this type of heads in the other models, which we also depict in Figure 4.9. The average attention weight placed on the question mark when the current token is an interrogative word is 98.5% for softmax, 97.0% for 1.5-entmax, and 99.5% for α-entmax.

Furthermore, we can examine sentences where some tendentially sparse heads become less so, thus identifying sources of ambiguity where the head is less confident in its prediction. An example is shown in Figure 4.10 where sparsity in the same head differs for sentences of similar length.

4.6 Subsequent Work

Correia et al. [2019] is part of an early line of work on transformers that sparked interest in making this model more efficient [Daras et al., 2020, Li et al., 2020, Merrill et al., 2021, Roy et al., 2021, *inter alia*], using sparsity to allow for transformers to use longer contexts more effectively [Jiang et al., 2020, Qiu et al., 2020, Sukhbaatar et al., 2021, *inter alia*], and being able to understand transformers better [You et al., 2020, Rogers et al., 2020, Pande et al., 2021, *inter alia*]. This transformer variant and α-entmax have also proved useful in medical applications [Guo et al., 2020, Yun et al., 2021].

Particularly connected to the present work, Treviso et al. [2022] tackles the quadratic complexity that remains in our approach, as we had focused more on the interpretability

[3] If the cluster has size 1, the score is the weight the token assigns to itself.

benefits of sparsity and not on its potential computational benefits. Treviso et al. [2022] propose *Sparsefinder*, a method that predicts in advance the sparsity pattern of α-entmax, by projecting queries and keys into a lower-dimensional space, turning our approach into a computationally efficient alternative to the original transformer.

Still on the topic of computational efficiency, Ji et al. [2021] focuses on inducing sparsity on the attention heads only at inference time to avoid unnecessary training of new models and wasting of computational resources. In order to achieve this, they propose a pruning and quantization technique that does not lead to decreased performance. To push the sparsity to the limit, Xu et al. [2021] propose a hard retrieval approach that is able to make each attention mechanism attend to only a single token. This approach also leads to similar performance to the original transformer but ends up having a 1.43 times faster decoding performance in translation tasks.

Regarding the concept of sparse transformers, Yun et al. [2020] unify several works that sparsify transformers in a single framework. This unified framework is constructed through conditions based on the sparsity pattern and the probability map. Once such conditions are satisfied, the authors prove that some sparse transformers (of which our approach is included) are universal approximators of any continuous sequence-to-sequence function. Furthermore, they show that when these sparse transformers have only $\mathcal{O}(n)$ connections (which is in contrast to the constant $\mathcal{O}(n^2)$ connections of dense transformers), they still hold the same universal approximation properties.

Zhang et al. [2021a] propose another approach to induce sparsity in each attention head of the transformer. In this work, instead of replacing the softmax with entmax, the authors replace it with the ReLU $= \mathsf{max}(0, x)$ activation. When compared to our approach, rectified linear attention (ReLA) obtains faster training and decoding time while achieving close performance in translation tasks. In their analysis, they followed our methodology and found that their method had higher JS divergence than our own, suggesting that attention heads using ReLA tend to be more diverse in where they attend at each layer. Furthermore, ReLA is also able to assign null attention for some queries,

that is, to effectively deactivate an attention head for a specific query, as it is able to assign zero values to all tokens. Their analysis shows that the rate at which null attention happens increases in deeper layers, which correlates with our own analysis that deeper layers contain less information. Moreover, Zhang et al. [2021b] achieves sparsity by pruning entire sections of the encoder outputs in the cross-attention of the decoder. This leads to a significant speed-up during decoding. It drops whole sections from the source encodings to speed up decoding, and the authors find that the tokens that end up being dropped are often uninformative, while the retained ones are relatively rare.

On the topic of interpretability and understanding the inner workings of transformers, inspired by our analysis (§4.5.2) along with other works [*e.g.*, Voita et al., 2019] that find that encoder self-attention heads learn fixed patterns, Raganato et al. [2020] fixes the attention pattern of all but one head in each layer, letting only a single head in each layer learn its attention. Those fixed patterns include the positional heads that focus on the current, previous, and next token and the BPE-merging head that we found. The parameter footprint of the model drastically decreases thanks to these simplifications, and the authors show empirically that translation quality does not drop significantly and even that, in low-resource scenarios, this approach improves translation performance.

4.7 Final Remarks and Chapter Summary

In the present chapter, we contributed a novel strategy for **learnable sparse** attention, and, in particular, for adaptively sparse transformers. We presented the first empirical analysis of transformers with sparse attention mappings (*i.e.*, entmax), showing potential in both translation accuracy as well as in the model interpretability.

In particular, we analyzed how the attention heads in the adaptively sparse transformer can specialize more and with higher confidence. Our adaptivity strategy relies only on gradient-based optimization, side-stepping costly per-head hyperparameter searches. Given the impact of our work, we believe that similar approaches will continue

to increase in the future, improving neural models' efficiency and **transparency**.

5

Efficient Marginalization of Discrete and Structured Latent Variables via Sparsity

Contents

In this chapter, we focus on the objective of making neural models more **compact**. We will propose a novel approach to train discrete and structured latent variable models that achieves an exact gradient unlike previous strategies in this field. Latent discrete and structured variables are of particular interest for the compactness of neural models since they can constrain the model and provide inductive bias. This family of latent variable models can also be easily semi-supervised, which can further increase the inductive bias and thus the compactness of the model.

The exactness of the gradient of our approach is achieved through explicitly doing the marginalization of Equation 2.17. As we shall see, this computation turns out to be efficient thanks to parameterizing the distribution over categories or structures with **sparsity**. Therefore, while in the previous chapter we have used sparsity for increased transparency, we will now use sparsity for increased efficiency. Albeit in a different way than in the previous chapter, this sparsity will too be **learned** over time, increasing as the training progresses due to the model being more confident on latent variable assignments.

This chapter is based on Correia, Niculae, Aziz, and Martins [2020].

5.1 Motivation

As discussed in §2.4.1, training with discrete latent variables can become challenging, due to the need to compute a gradient of a large sum over all possible latent variable assignments, with each term itself being potentially expensive (Equation 2.17). This challenge is typically tackled by estimating the gradient with Monte Carlo methods [MC; Mohamed et al., 2019], which rely on sampling estimates. The two most common strategies for MC gradient estimation are the score function estimator [SFE; Rubinstein, 1976, Paisley et al., 2012], which suffers from high variance, or surrogate methods that rely on the continuous relaxations, like straight-through [Bengio et al., 2013] or Gumbel-Softmax [Maddison et al., 2017, Jang et al., 2017], which potentially reduce variance but introduce bias and modeling assumptions.

In this work, we take a step back and ask: Can we avoid sampling entirely, and instead, deterministically evaluate the sum in Equation 2.17 with less computation? To answer affirmatively, we propose an alternative method to train these models by parameterizing the discrete distribution with **sparse mappings** — sparsemax [Martins and Astudillo, 2016] and two structured counterparts, SparseMAP [Niculae et al., 2018a] and a novel mapping top-k sparsemax. Sparsity implies that some assignments of the latent variable are entirely ruled out. This leads to the corresponding terms in the sum evaluating trivially to zero, allowing us to disregard potentially expensive computations.

Contributions. We introduce a general strategy for learning discrete latent variable models that hinges on learning a sparse distribution over the possible assignments. In the unstructured categorical case, our strategy relies on the sparsemax activation function, presented in §5.3, while in the structured case we propose two strategies, SparseMAP and top-k sparsemax, presented in §5.4. Unlike existing approaches, our strategies involve neither MC estimation nor any relaxation of the discrete latent variable to the continuous space. We demonstrate our strategy in three different applications: a semi-supervised generative model, an emergent communication game, and a bit-vector variational autoencoder. We provide a thorough analysis and comparison to MC methods, and — when feasible — to exact marginalization. Our approach is consistently a top performer, combining the accuracy and robustness of exact marginalization with the efficiency of single-sample estimators.

5.2 Previous Work

Differentiable sparse mappings. There has been recent interest in applying sparse mappings (see §2.3) of discrete distributions in deep discriminative models [Martins and Astudillo, 2016, Niculae et al., 2018a, Niculae and Blondel, 2017, Peters et al., 2019, Niculae et al., 2018b], attention mechanisms [Malaviya et al., 2018, Shao et al., 2019, Maruf et al., 2019, Correia et al., 2019], and in topic models [Cao, 2019]. Our work focuses on

the parameterization of distributions over latent variables with sparse mappings, on the computational advantage to be gained by sparsity, and on the contrast between our novel training method and common sampling-based methods.

Reducing sampling noise. The sampling procedure found in SFE is a great source of variance in models that use it. To reduce this variance, many works have proposed baselines [Williams, 1992, Gu et al., 2016, Wang et al., 2013]. VIMCO [Mnih and Rezende, 2016] is a multi-sample estimator which exploits variance reduction via input-dependent baselines as well as a lower bound on marginal likelihood which is tighter than the ELBO [Burda et al., 2016]. The number of samples in VIMCO is a hyperparameter that stays fixed throughout training. Our methods, in contrast, may take several decoder calls initially, but that number automatically decreases over time as training progresses. While baselines must be independent of the sample for which we assess the score function, exploiting correlation in the downstream losses of dependent samples holds potential for further variance reduction. These are known as control variates [Greensmith et al., 2004]. REBAR [Tucker et al., 2017] exploits a continuous relaxation to obtain a dependent sample and uses the downstream loss assessed at the relaxed sample to define a control variate. RELAX [Grathwohl et al., 2018], instead, learns to predict the downstream loss of the relaxed sample with an auxiliary network. In contrast, sparse marginalization works for any factorization where a primitive for 1-best (or k-best) enumeration is available and takes no additional parameters nor additional optimization objectives. Another line of work approximates argmax gradients by perturbed finite differences Lorberbom et al. [2019], Vlastelica et al. [2020]; this requires the same computation primitive as our approach but is always biased. ARM [Yin and Zhou, 2019] is a control variate based on antithetic samples [Owen, 2013]: it does not require relaxation nor additional parameters, but it only applies to factorial Bernoulli distributions. Closest to our work are variance reduction techniques that rely on partial marginalization, typically of the top-k assignments to the latent variable [Liu et al., 2019, Kool et al., 2020]. These methods show improved performance and variance reduction, but require rejection sampling,

which can be challenging in structured problems.

5.3 Efficient Marginalization via Sparsity

As discussed in §2.4, the challenge of computing the exact expectation in Equation 2.17 is linked to the need to compute a sum with a large number of terms. This holds when the probability distribution over latent assignments is *dense* (*i.e.*, every assignment $z \in \mathcal{Z}$ has non-zero probability), which is indeed the case for most parameterizations of discrete distributions. Our proposed methods hinge on *sparsifying* this sum.

Take the example where $\mathcal{Z} = \{1, \ldots, K\}$, with a neural network predicting from x to a K-dimensional vector of real-valued scores $\boldsymbol{s} = \boldsymbol{g}(x; \theta)$, such that s_z is the score of z.[1] The traditional way to obtain the vector $\boldsymbol{\xi}$ parameterizing $\pi(z|x; \theta)$ is with the softmax transform (*i.e.*, $\boldsymbol{\xi} = \mathsf{softmax}(\boldsymbol{s})$). Since this gives $\pi(z|x; \theta) \propto \mathsf{exp}(s_z)$, the expectation in Equation 2.17 depends on $\ell(x, z; \theta)$ for every possible z.

We rethink this standard parameterization, proposing a **sparse** mapping from scores to the simplex. In particular, we substitute the softmax activation function by sparsemax [Martins and Astudillo, 2016], described in §2.3.

Our main insight is that with a sparse parameterization of π, we can compute the expectation in Equation 2.17 evaluating $\ell(x, z; \theta)$ only for assignments $z \in \bar{\mathcal{Z}} := \{z : \pi(z|x, \theta) > 0\}$. This leads to a powerful alternative to MC estimation, which requires fewer than $|\mathcal{Z}|$ evaluations of ℓ, and which strategically — yet deterministically — selects which assignments $\bar{\mathcal{Z}}$ to evaluate ℓ on. Empirically, our analysis in §5.5 reveals an adaptive behavior of this sparsity-inducing mechanism, performing more loss evaluations in early iterations while the model is uncertain, and quickly reducing the number of evaluations, especially for unambiguous data points. This is a notable property of our learning strategy: In contrast, MC estimation cannot decide when an ambiguous data point may require more sampling for accurate estimation; and directly evaluating Equation 2.17

[1] Not to be confused with "score function," as in SFE, which refers to the gradient of the log-likelihood.

with the dense $\boldsymbol{\xi}$ resulting from a softmax parameterization never reduces the number of evaluations required, even for simple instances.

5.4 Structured Latent Variables

As discussed in §2.4.2, many interesting models can include latent variables that exist in a set of combinatorial size. While it may be tempting to consider using sparsemax to avoid the expensive sum in the exact expectation of Equation 2.17, this is prohibitive too: solving the problem in Equation 2.8 still requires explicit manipulation of the large vector $\boldsymbol{s} \in \mathbb{R}^{|\mathcal{Z}|}$, and even if we could avoid this, in the worst case ($\boldsymbol{s} = \mathbf{0}$) the resulting sparsemax distribution would still have exponentially large support. Fortunately, we show next that it is still possible to develop sparsification strategies to handle the combinatorial explosion of $\mathcal{Z}$ in the structured case. We propose two different methods to obtain a sparse distribution $\boldsymbol{\xi}$ supported only over a bounded-size subset of $\mathcal{Z}$: top-k sparsemax (§5.4.1) and SparseMAP (§5.4.2).

5.4.1 Top-k Sparsemax

Recall that the sparsemax operator (Equation 2.8) is simply the Euclidean projection onto the $|\mathcal{Z}|$-dimensional probability simplex. While there is a propensity for sparsity, there is no upper bound on the number of non-zeros of the resulting distribution. When $\mathcal{Z}$ is large, one possibility is to add a cardinality constraint $\|\boldsymbol{\xi}\|_0 \leq k$ for some prescribed $k \in \mathbb{N}$. The resulting problem becomes

$$\mathsf{sparsemax}_k(\boldsymbol{s}) := \operatorname*{argmin}_{\boldsymbol{\xi} \in \triangle^{|\mathcal{Z}|-1},\, \|\boldsymbol{\xi}\|_0 \leq k} \|\boldsymbol{\xi} - \boldsymbol{s}\|_2^2, \tag{5.1}$$

which is known as a *sparse projection onto the simplex* and has been studied in detail by Kyrillidis et al. [2013] and used to smooth structured prediction losses [Pillutla et al., 2018, Blondel et al., 2020]. Remarkably, while this is a non-convex problem, its solution

$\boldsymbol{\xi}^\star$ can be written as a composition of two functions: a top-k operator $\mathsf{top}_k : \mathbb{R}^{|\mathcal{Z}|} \to \mathbb{R}^{|\mathcal{Z}|}$, which returns a vector identical to its input but where all the entries not among the k largest ones are masked out (set to $-\infty$), and the k-dimensional sparsemax operator.

Formally, $\mathsf{sparsemax}_k = \mathsf{sparsemax}(\mathsf{top}_k(s))$. Being a composition of operators, its Jacobian becomes a product of matrices and hence simple to compute.[2]

To apply the top-k sparsemax to a large or combinatorial set $\mathcal{Z}$, all we need is a primitive to compute the top-k entries of s — this is available for many structured problems (for example, sequential models via k-best dynamic programming) and, when $\mathcal{Z}$ is the set of joint assignments of D discrete binary variables, it can be done with a cost $\mathcal{O}(kD)$.

After enumerating this set, we parameterize $\pi(z|x;\theta)$ by applying sparsemax to that top-k, with a computational cost $\mathcal{O}(k)$. Note that **this method is identical to sparsemax whenever** $\|\mathsf{sparsemax}(s)\|_0 \leq k$: if during training the model learns to assign a sparse distribution to the latent variable, we are effectively using a sparsemax parameterization as presented in §5.3 with cheap computation. In fact, the solution of Equation 5.1 gives us a certificate of optimality whenever $\|\boldsymbol{\xi}^\star\|_0 < k$.

5.4.2 SparseMAP

A second possibility to obtain efficient summation over a combinatorial space without imposing any constraints on $\ell(x, z; \theta)$ is to use SparseMAP [§2.3.3; Niculae et al., 2018a,b]. Due to the properties of SparseMAP, assessing the expectation in Equation 2.17 only requires evaluating $|\tilde{\mathcal{Z}}| = \mathcal{O}(D)$ terms.

5.5 Experimental Analysis

We next demonstrate the applicability of our proposed strategies by tackling three tasks: a deep generative model with semi-supervision (§5.5.1), an emergent communication two-

[2] the Jacobian of top_k is a diagonal matrix whose diagonal is a multi-hot vector indicating the top-k elements of s

player game over a discrete channel (§5.5.2), and a variational auto-encoder with latent binary factors (§5.5.3).

We follow the experimental procedures described in [Liu et al., 2019] and [Lazaridou et al., 2017] for §5.5.1 and §5.5.2, respectively. We describe the most relevant training details and key differences in architectures when applicable. For other implementation details that we do not mention here, we refer the reader to the works referenced above. For all Gumbel baselines, we relax the sample into the continuous space but assume a discrete distribution when computing the entropy of $\pi(z|x;\theta)$, as suggested as one implementation option in Maddison et al. [2017]. Our code is publicly available[3] and was largely inspired by the structure and implementations found in EGG [Kharitonov et al., 2019], having been built upon it.

5.5.1 Semi-Supervised Variational Auto-Encoder

We consider the semi-supervised Variational Auto-Encoder (VAE) of Kingma et al. [2014], which models the joint probability

$$p_{XZH}(z, h, x|\phi) = p_Z(z)p_H(h)p_{X|ZH}(x|z, h)\,,$$

where x is an observation (*e.g.*, an MNIST image), h is a continuous latent variable with a n-dimensional standard Gaussian prior, and z is a discrete random variable with a uniform prior over K categories.

The semi-supervised objective is

$$L_{\mathcal{D}}(\phi) = -\log p_X(x|\phi) - \log p_{XZ}(x, z|\phi) + \mathcal{R}(\phi), \tag{5.2}$$

where $\mathcal{R}(\phi)$ is a regularizer we will specify below, and $\log p_X(x|\phi)$ and $\log p_{XZ}(x, z|\phi)$ is the log-likelihood of the unsupervised and supervised components, respectively.

[3] https://github.com/deep-spin/sparse-marginalization-lvm

For the unsupervised component of the loss, note that the marginal likelihood

$$p_X(x|\phi) = \sum_{z=1}^{K} \int_h p_{X|ZH}(x|z, h; \phi) p_H(h) p_Z(z) \, \mathrm{d}h$$

is intractable, due to the marginalization of $h \in \mathbb{R}^n$. For the same reason, the marginal likelihood of the supervised component of the loss $p_{XZ}(x, z|\phi)$ is also intractable. Additionally, on the unsupervised component, for a fixed h (*e.g.*, sampled), marginalizing z requires K calls to the decoder, which can be costly depending on its architecture.

To circumvent the need for the marginal likelihood, Kingma et al. [2014] use variational inference [Jordan et al., 1999] with an approximate posterior $\pi(z|x; \theta_\pi) q(h|z, x; \lambda)$. This trains a classifier $\pi(z|x; \theta_\pi)$ along with the generative model. In Kingma et al. [2014], h is sampled with the reparameterization trick [Kingma and Welling, 2014, Rezende et al., 2014], and the expectation over z is computed in closed-form, that is, assessing all K terms of the sum for a sampled h. Under the notation in §2.4, we let $\theta_\ell = \{\lambda, \phi\}$ and define $\pi(z|x; \theta_\pi) \coloneqq q(z|x; \lambda)$ and the loss in Equation 5.2 turns into

$$L_{\mathcal{D}}(\theta) = \sum_{z' \in \mathcal{Z}} \pi(z'|x; \theta_\pi) \ell_U(x, z'; \theta_\ell) + \log \pi(z|x; \theta_\pi) \ell_L(x, z; \theta_\ell) + \mathcal{R}(\theta), \tag{5.3}$$

where for the unsupervised component

$$\begin{aligned} \ell_U(x, z; \theta_\ell) \coloneqq & - \mathbb{E}_{q(h|z,\lambda)} \left[\log p_{X|ZH}(x|z, h; \phi) \right] \\ & - \log \frac{p_Z(z)}{\pi(z|x; \theta_\pi)} \\ & + \mathrm{KL} \left[q(h|x, z; \lambda) \parallel p_H(h) \right], \end{aligned}$$

which turns Equation 2.17 into the (negative) evidence lower bound (ELBO). For the

supervised component in Equation 5.2 we let

$$\begin{aligned}\ell_L(x, z; \theta_\ell) \coloneqq &- \mathbb{E}_{q(h|z,\lambda)} \left[\log p_{X|ZH}(x|z, h; \phi)\right] \\ &- \log p_Z(z) \\ &+ \mathsf{KL}\left[q(h|x, z; \lambda) \parallel p_H(h)\right].\end{aligned}$$

Finally, to complete the specification of Equation 5.3, we let $\mathcal{R}(\theta) = -\log \pi(z|x; \theta_\pi)$ when z is supervised in order to train the classifier on the supervised data as well. To update $q(h|x, z; \lambda)$, we use the reparameterization trick to obtain gradients through a sampled h. For $\pi(z|x; \theta_\pi)$, we may still explicitly marginalize over each possible assignment of z, but this has a multiplicative cost on K. As an alternative, we parameterize $\pi(z|x, \theta_\pi)$ with a sparse mapping, comparing it to the original formulation and with stochastic gradients based on SFE and continuous relaxations of z.

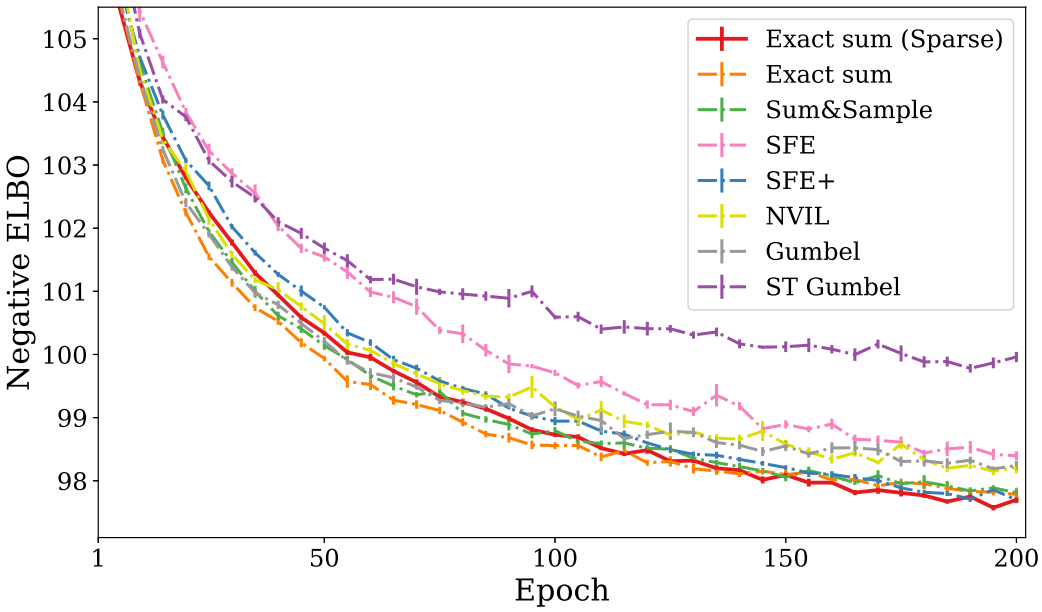


Figure 5.1: Learning curves on the test set for semi-supervised VAE on MNIST.

Data and architecture. We evaluate this model on the MNIST dataset [LeCun et al., 1998], using 10% of labeled data, treating the remaining data as unlabeled. MNIST consists of 28×28 gray-scale images of hand-written digits. It contains 60K data points for

Method	Accuracy (%)	Decoder calls
Monte Carlo		
SFE	94.75± .002	1
SFE+	96.53± .001	2
NVIL	96.01± .002	1
Sum&Sample	96.73± .001	2
Gumbel	95.46± .001	1
ST Gumbel	86.35± .006	1
Marginalization		
Dense	96.93± .001	10
Sparse (proposed)	96.87± .001	1.01± 0.01

Table 5.1: Average test results and standard errors over 10 runs for semi-supervised VAE on MNIST.

training and 10K for testing. We perform model selection on the last 10K data points of the training split. In this experiment, the classification network consists of three fully connected hidden layers of size 256, using ReLU activations. The generative and inference network both consist of one hidden layer of size 128, also with ReLU activations. The multivariate Gaussian has 8 dimensions and its covariance is diagonal. For all models we have chosen the learning rate based on the best ELBO on the validation set, doing a grid search (5e-5, 1e-4, 5e-4, 1e-3, 5e-3). The accuracy shown in Table 5.1 is the test accuracy taken after the last epoch of training. The temperature of the Gumbel models was annealed according to $\tau = \max(0.5, -rt)$, where t is the global training step. For these models, we also did a grid search over r (1e-5, 1e-4) and over the frequency of updating τ every (500, 1000) steps. Optimization was done with Adam. For our method, in the labeled loss component of the semi-supervised objective, we used the sparsemax loss [Martins and Astudillo, 2016]. Following Liu et al. [2019], we pre-train the network with only labeled data before training with the whole training set. Likewise, for our method, we pre-trained the network on the sparsemax loss and every other method with the Negative Log-Likelihood loss. Each model was trained for 200 epochs.

Comparisons. Our proposal's key ingredient is sparsity, which permits exact marginalization and a deterministic gradient. To investigate the impact of sparsity alone, we report

a comparison against the exact marginalization over the entire support of $\mathcal{Z}$ using a dense softmax parameterization. To investigate the impact of deterministic gradients, we compare to stochastic gradients:

- SFE with a moving average baseline;
- SFE with a self-critic baseline [SFE+; Rennie et al., 2017], that is, we use $\log p_{X|ZH}(x|z', h; \phi)$ as baseline, where $z' \sim \pi(z|x; \theta_\pi)$ is an independent sample;
- NVIL [Mnih and Gregor, 2014] with a learned baseline (we train an MLP to predict the learning signal by minimizing mean squared error); [4]
- Sum-and-sample [Liu et al., 2019];
- Gumbel-Softmax, which relaxes the random variable to the interior of the simplex;
- ST Gumbel-Softmax, which discretizes the relaxation in the forward pass, but ignores the discretization function in the backward pass.[5]

Results and discussion. In Figure 5.1, we see that our proposed sparse marginalization approach performs just as well as its dense counterpart, both in terms of ELBO and accuracy. However, by inspecting the number of times each method calls the decoder for assessments of $p_{X|ZH}(x|z, h; \phi)$, we can see that the effective support of our method is much smaller — sparsemax-parameterized posteriors get very confident, and mostly require one, and sometimes two, calls to the decoder. Regarding the Monte Carlo methods, the continuous relaxation done by Gumbel-Softmax underperformed all the other methods, except for SFE with a moving average. While SFE+ and Sum&Sample are very strong performers, they will always require throughout training the same number of calls to the decoder (in this case, two). On the other hand, sparsemax makes a small number

[4] NVIL and SFE+ are similar, the difference being that the baseline in SFE+ does not require additional parameters nor does it introduce additional objectives.

[5] For Gumbel-Softmax (with and without ST), we follow Jang et al. [2017] and substitute $\mathsf{KL}(\pi(z|x; \theta_\pi) \| p_Z(z))$ in the ELBO by the KL divergence of $\mathsf{Categorical}(\mathsf{softmax}(s))$ from a discrete uniform prior. Strictly speaking, this means the objective is not a proper ELBO and its relationship to an ELBO is unclear [Maddison et al., 2017, Appendix C.2].

of decoder calls not due to a choice in hyperparameters but thanks to the model converging to only using a small support, which can endow this method with a lower number of computations as it becomes more confident.

5.5.2 Emergent Communication Game

Emergent communication studies how two agents can develop a communication protocol to solve a task collaboratively [Kirby, 2002]. Recent work used neural latent variable models to train these agents via a "collaborative game" between them [Lazaridou et al., 2017, Havrylov and Titov, 2017, Jorge et al., 2016, Foerster et al., 2016, Sukhbaatar et al., 2016]. In Lazaridou et al. [2017], one of the agents, the *sender*, sees an image v_y and sends a single symbol message z chosen from the *vocabulary* set $\mathcal{Z}$ to the other agent, the *receiver*, who needs to choose v_y out of a set of images $\mathcal{V} = \{v_1, \ldots, v_C\}$.[6] They found that the messages communicated this way can be correlated with broad object properties amenable to interpretation by humans. In our framework of Equation 2.17, we let $x = (\mathcal{V}, y)$ and define

$$\ell(x, z; \theta) \coloneqq -\log p_{X|Z}(y|\mathcal{V}, z; \theta_\ell)$$

and

$$\pi(z|x; \theta) \coloneqq p_{Z|X}(z|v_y; \theta_\pi)\,,$$

where $p_{X|Z}(y|\mathcal{V}, z; \theta_\ell)$ corresponds to the sender and $p_{Z|X}(z|v_y; \theta_\pi)$ to the receiver. Following Lazaridou et al. [2017], we add an entropy regularization of $\pi(z|x; \theta)$ to the loss, with a coefficient as an hyperparameter [Mnih et al., 2016].

Data and architecture. In this application, we closely followed the experimental procedure described by Lazaridou et al. [2017] with a few key differences. The architecture of

[6] Lazaridou et al. [2017] lets the sender see the full set $\mathcal{V}$. However, we follow Havrylov and Titov [2017] in showing only the correct image v_y to the sender, making the game harder, as the message z needs to encode a good "description" of v_y instead of encoding only its differences from $\mathcal{V} \setminus \{v_y\}$.

the sender and the receiver is identical with the exception that the sender does not take as input the distracting images along with the correct image — only the correct image. To make the game even harder, we increase the collection of images $|\mathcal{V}|$ as suggested by Havrylov and Titov [2017]; in our experiments, we increase it from 2 to 16. and the vocabulary of the sender was increased to 256. The hidden size and embedding size were also increased to 512 and 256, respectively. We did a grid search on the learning rate (0.01, 0.005, 0.001) and entropy regularizer (0.1, 0.05, 0.01) and chose the best configuration for each model on the validation set based on the communication success. For the Gumbel models, we applied the same schedule and grid search to the temperature as described in §5.5.1. All models were trained with the Adam optimizer, with a batch size of 64 and during 200 epochs. We choose the vocabulary of the sender to be 256, the hidden size to be 512 and the embedding size to be 256. All methods are trained for 500 epochs. The data used by Lazaridou et al. [2017] is a subset of ImageNet containing 463,000 images, chosen by sampling 100 images from 463 base-level concepts. The images are then applied a forward-pass through the pre-trained VGG ConvNet [Simonyan and Zisserman, 2015] and the representations at the second-to-last fully connected layer are saved to use as input to the sender/receiver.

Comparisons. We compare our method to stochastic gradient estimators as well as exact marginalization under a dense softmax parameterization of $p_{Z|X}(z|v_y;\theta_\pi)$. Again, we have unbiased (SFE with moving average baseline, SFE+, and NVIL) and biased (Gumbel-Softmax and ST Gumbel-Softmax) estimators. For SFE we also experiment using a 0/1 loss.

Results and discussion. Table 5.2 shows the communication success (accuracy of the receiver at picking the correct image v_y). While the communication success for $|\mathcal{V}| = 2$ in Lazaridou et al. [2017] was close to perfect, we see that increasing $|\mathcal{V}|$ to 16 makes this game much harder to sampling-based approaches. Only the models that do explicit marginalization achieve close to perfect communication in the test set. However, as $\mathcal{L}$

Method	Communication success (%)	Decoder calls
Monte Carlo		
SFE (NLL)	33.05 ± 2.84	1
SFE (0/1)	55.36 ± 2.92	1
SFE+ (0/1)	44.32 ± 2.72	2
NVIL	37.04 ± 1.61	1
Gumbel	23.51 ± 16.19	1
ST Gumbel	27.42 ± 13.36	1
Marginalization		
Dense	93.37 ± 0.42	256
Sparse (proposed)	93.35 ± 0.50	3.13 ± 0.48

Table 5.2: Emergent communication success test results, averaged across 10 runs. Random guess baseline 6.25%.

increases, marginalizing with a softmax parameterization gets computationally more expensive, as it requires $|\mathcal{Z}|$ forward and backward passes on the receiver. Unlike softmax, the model trained with sparsemax gives very small support, requiring on average only 3 decoder calls. In fact, sparsemax starts off dense while exploring, but quickly becomes very sparse (Figure 5.2).

5.5.3 Bit-Vector Variational Auto-Encoder

As described in §5.4, combinatorial interactions and constraints will make $\mathcal{Z}$ exponentially large. In this section, we study the illustrative case of encoding images into a binary codeword z, by training a latent bit-vector variational auto-encoder [Jang et al., 2017, Mnih and Gregor, 2014]. One approach for parameterizing the approximate posterior is to use a Gibbs distribution, decomposable as a product of independent Bernoulli,

$$q(z|x;\lambda) \propto \exp(\langle \boldsymbol{a}_z, \boldsymbol{t} \rangle) = \prod_{i=1}^{D} q(z_i|x;\lambda)\,,$$

with each z_i being a Bernoulli with parameter t_i, and D being the size of the bit-vector. While marginalizing over all the possible z is intractable, drawing samples can be done efficiently by sampling each component independently, and the entropy has a closed-form

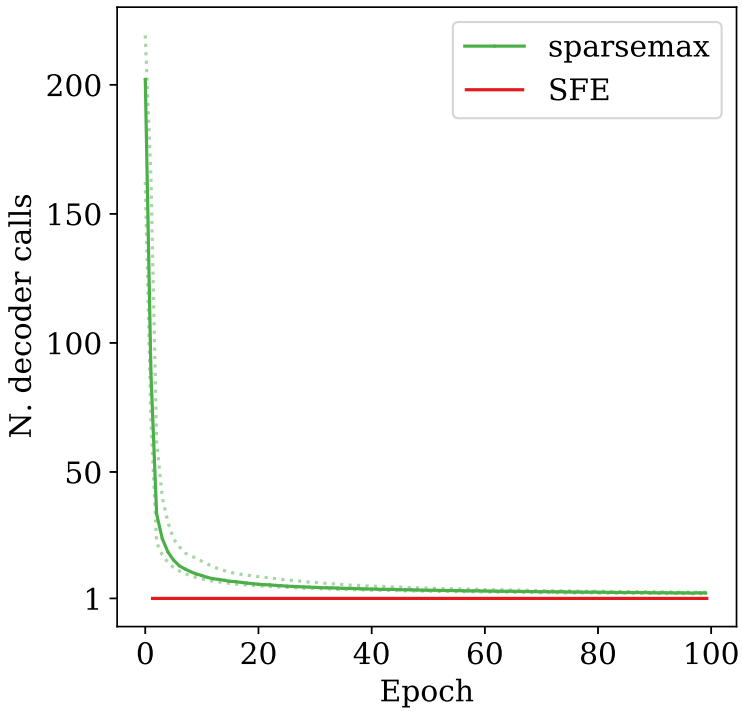


Figure 5.2: Median decoder calls per epoch during training time with 10 and 90 percentiles in dotted lines by sparsemax.

expression. This efficient sampling and entropy computation relies on an independence assumption; in general, we may not have access to such efficient computation.

Training this VAE to minimize the negative ELBO corresponds to

$$\ell(x, z; \theta_\ell) \coloneqq -\log \frac{p_{XZ}(x, z|\phi)}{q(z|x, \lambda)};$$

we use a uniform prior $p_Z(z) = 1/|\mathcal{Z}| = 1/2^D$. This objective does not constrain $\pi(z|x; \theta_\pi) \coloneqq q(z|x, \lambda)$ to the Gibbs parameterization, and thus to apply our methods we will differ from it.

Top-k sparsemax parameterization. As pointed out in §5.4, we now cannot explicitly handle the sparsemax mapping $\boldsymbol{\xi} = \mathsf{sparsemax}(\boldsymbol{s})$, as it involves a vector of dimension 2^D. However, given $\boldsymbol{t}$, we can efficiently find the k largest configurations in time $\mathcal{O}(kD)$, with the procedure described in §5.4.1, and thus we can evaluate $\mathsf{sparsemax}_k(\boldsymbol{s})$ efficiently.

SparseMAP parameterization. Another sparse alternative to the intractable structured sparsemax, as discussed in §5.4, is SparseMAP. In this case, we compute an optimal distribution $\boldsymbol{\xi}$ using the active set algorithm of Niculae et al. [2018a], by using a maximization oracle which can be computed in $\mathcal{O}(D)$:

$$\operatorname*{argmax}_{z} \langle \boldsymbol{a}_z, \boldsymbol{t} \rangle = z^\star \quad \text{s.t.} \quad [\boldsymbol{a}_{z^\star}]_i = \begin{cases} 1, & t_i \geq 0 \\ 0, & t_i < 0 \end{cases}.$$

Since SparseMAP can handle structured problems, we also experimented with adding a *budget constraint* to SparseMAP: this is done by adding a constraint $\|\boldsymbol{z}\|_1 \leq B$, where $B \leq D$; we used $b = \frac{D}{2}$. The budget constraint ensures the images are represented with sparse codes, and the maximization oracle can be computed in $\mathcal{O}(D \log D)$. This is done by sorting the Bernoulli scores and selecting the entries among the top-B which have a positive score.

We stress that, with both top-k sparsemax and SparseMAP parameterizations, z does not decompose into a product of independent binary variables: this property is specific to the Gibbs parameterization. However, since these new approaches induce a very sparse approximate posterior q, we may compute the terms $\mathbb{E}_{q(z|x;\lambda)}[\log p_{X|Z}(x|z, \phi)]$ and $\mathbb{E}_{q(z|x;\lambda)}[\log q(z|x; \lambda)]$ explicitly.

Data and architecture. We use Fashion-MNIST [Xiao et al., 2017], consisting of 28×28, 256-level gray-scale images of clothes. It contains 60K data points for training and 10K data points for testing. We perform model selection on the last 10K data points of the training split. The decoder uses an independent categorical distribution for each pixel, $p_{X|Z}(x|z; \phi) = \prod_{i=1}^{28} \prod_{j=1}^{28} p_{X|Z}(x_{ij}|z; \phi)$. For top-$k$ sparsemax, we choose $k = 10$. We have set the generative and inference network to consist of one hidden layer with 128 nodes, using ReLU activations. We have searched a learning rate by doing a grid search (0.0005, 0.001, 0.002) and have chosen the model based on the ELBO performance on the validation set. For the Gumbel models, we applied the same schedule and grid

Method	$D = 32$	$D = 128$
Monte Carlo		
SFE	3.74	3.77
SFE+	3.61	3.59
NVIL	3.65	3.60
Gumbel	3.57	3.49
ST Gumbel	3.53	3.55
Marginalization		
Top-k sparsemax	3.62	3.61
SparseMAP	3.72	3.67
SparseMAP (w/ budget)	3.64	3.66

Table 5.3: Test results for Fashion-MNIST. NLL in bits/dim (lower, the better).

search to the temperature as described in §5.5.1. We used the Adam optimizer.

Comparisons. This time, exact marginalization under a dense parameterization of $q(z|x;\lambda)$ is truly intractable, so we can only compare our method to stochastic gradient estimators. We have unbiased SFE-based estimators (SFE with moving average baseline, SFE+, and NVIL), and biased reparameterized gradient estimators (Gumbel-Softmax and ST Gumbel-Softmax). As there is no supervision for the latent code, we cannot compare the methods in terms of accuracy or task success. Instead, we display the trained models in the rate-distortion (RD) plane [Alemi et al., 2018][7] and also report bits-per-dimension of x, estimated with importance sampling, on held-out data.

Bits-per-dimension is the negative logarithm of the marginal likelihood normalized per number of pixels in the image, thus we need to assess or estimate the marginal likelihood of observations. For dense parameterizations, the usual option is importance sampling (IS) using the trained approximate posterior as importance distribution: *i.e.*,

$$\log p_X(x|\phi) \overset{\text{IS}}{\approx} \log\left(\frac{1}{S}\sum_{s=1}^{S}\frac{p_{XZ}(z^{(s)}, x|\phi)}{q(z^{(s)}|x;\lambda)}\right)$$

with $z^{(s)} \sim q(z|x;\lambda)$. The result is a stochastic lower bound which converges to the true

[7] Distortion is the expected value of the reconstruction negative log-likelihood, while the rate is the average KL divergence from the prior to the approximate posterior.

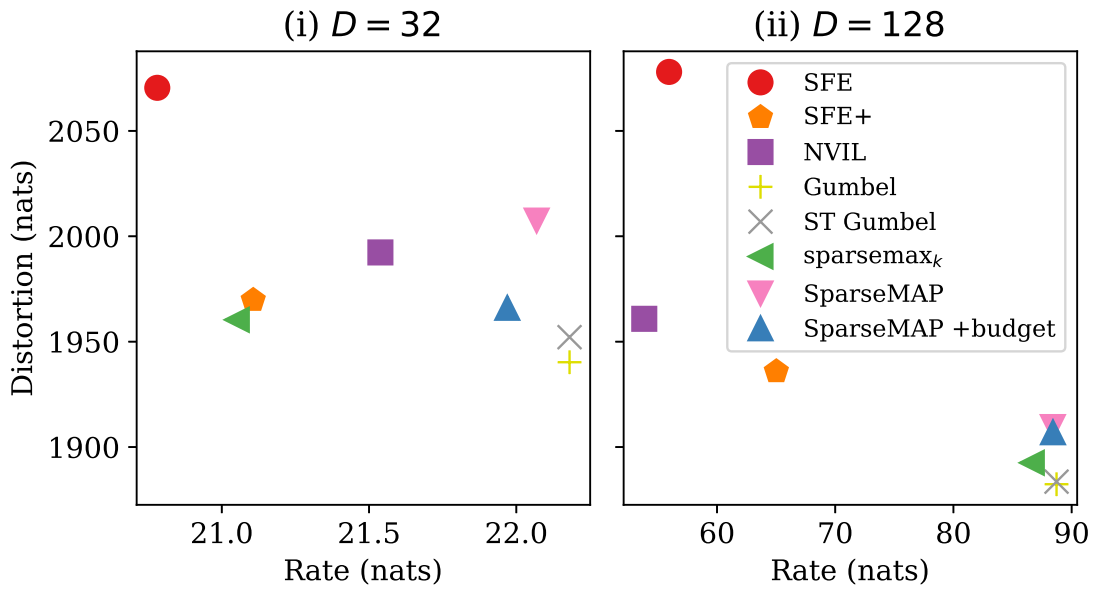


Figure 5.3: Test results for Fashion-MNIST. RD plots (the closer to the lower right corner, the better).

log-marginal in the limit as $S \to \infty$. With a sparse posterior approximation, we can split the marginalization

$$\log p_X(x|\phi) = \log\left(\sum_{z\in\bar{\mathcal{Z}}} p_Z(z)p_{X|Z}(x|z;\phi) + \sum_{z\in\mathcal{Z}\setminus\bar{\mathcal{Z}}} p_Z(z)p_{X|Z}(x|z;\phi)\right)$$

into one part that handles outcomes in the support $\bar{\mathcal{Z}}$ of the sparse posterior approximation and another that handles the outcomes in the complement set $\mathcal{Z} \setminus \bar{\mathcal{Z}}$. We compute the first part exactly and estimate the second part via rejection sampling from $p_Z(z)$.

Results and discussion. Table 5.3 shows an importance sampling estimate (1024 samples per test example were taken) of the negative log-likelihood for the several methods, together with the converged values of each method in the RD plane in Figure 5.3. Both show results for which the bit-vector has dimensionality $D = 32$ and $D = 128$. Regarding the estimated negative log-likelihood, our methods exhibit increased performance when compared to SFE, and top-k sparsemax is competitive with the remaining unbiased estimators. However, in the RD plane, both our methods show comparable performance

to SFE+ and NVIL for $D = 32$, but for $D = 128$ all of our methods have a significantly higher rate and lower distortion than any unbiased estimator, suggesting a better fit of $p_X(x|\phi)$ [Alemi et al., 2018]. In Figure 5.4, we can observe the training progress in number of calls to $p_{X|Z}(x|z;\phi)$ for the models with 32 and 128 latent bits, respectively. While $\mathsf{sparsemax}_k$ introduces bias towards the most probable assignments and may discard outcomes that sparsemax would assign a non-zero probability to, as training progresses distributions may (or tend to) be sufficiently sparse and this mismatch disappears, making the gradient computation exact.

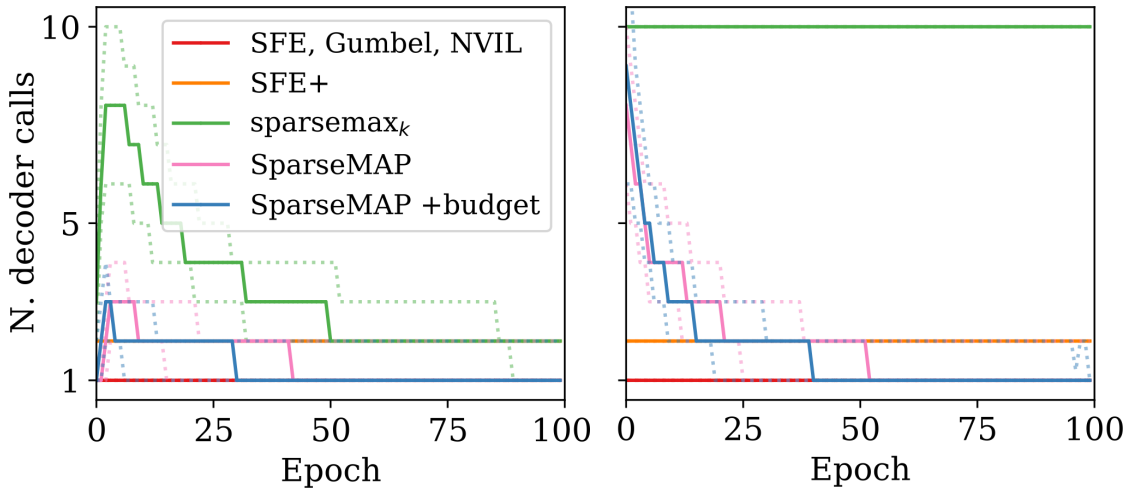
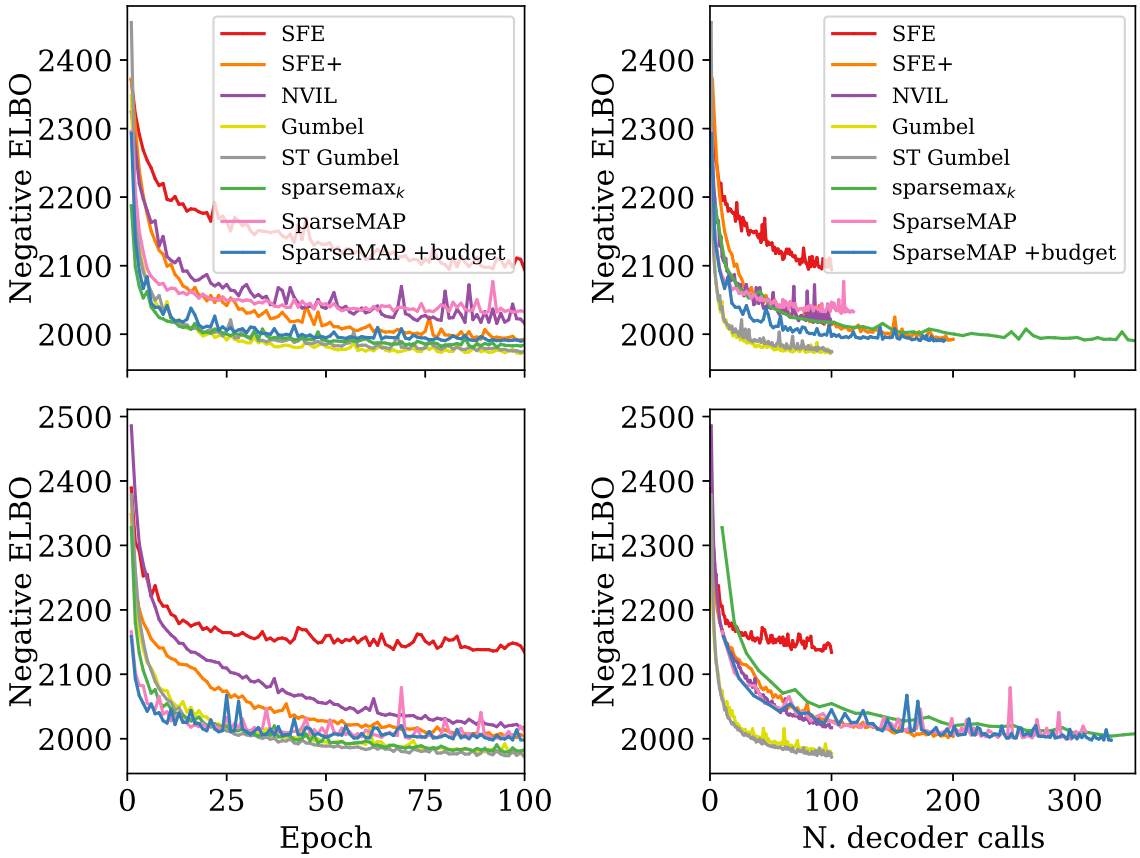


Figure 5.4: Bit vector VAE median and quartile decoder calls per epoch, $D = 32$ (left) / $D = 128$ (right).

Remarkably, this happens for $D = 32$ — the support of $\mathsf{sparsemax}_k$ is smaller than k, giving the true gradient to $q(z|x;\lambda)$ for most of the training. This no longer happens for $D = 128$, for which it remains with full support throughout, due to the much larger search space. On the other hand, SparseMAP solutions become very sparse from the start in both cases, while still obtaining good performance. There is, therefore, a trade-off between the solutions we propose: on one hand, $\mathsf{sparsemax}_k$ can become exact with respect to the expectation in Equation 2.17, but it only does so if the chosen k is suitable to the difficulty of the model; on the other hand, SparseMAP may not offer an exact gradient to $q(z|x;\lambda)$, but its performance is very close to $\mathsf{sparsemax}_k$ and its higher propensity for sparsity gifts it with less computation. Figure 5.5 shows the downstream loss (ELBO)

over epochs and over the median number of decoder calls per epoch. The plots on Figure 5.5b show how our methods have comparable computational overhead to sampling approaches. Oftentimes, our methods could have been trained in fewer epochs to obtain the same performance as the sampling estimators have for 100 epochs.



(a) Neg. ELBO over training epochs.

(b) Neg. ELBO over decoder calls.

Figure 5.5: Performance on the validation set for the experiment in §5.5.3, $D = 32$ (top) / $D = 128$ (bottom). For $D = 32$, top-k sparsemax continues until a total of 561 median decoder calls, and for $D = 128$ it continues until a total of 998.

Concerning relaxed estimators, note that the reconstruction loss is computed given a continuous sample, rather than a discrete one, allowing it more flexibility to directly reduce distortion and potentially explaining why it does well in that regard. Moreover, the rate of the relaxed model is unknown,[8] and instead we plot the rate as if z was given discrete treatment, which, although common practice, makes comparisons to other esti-

[8] Estimating it would require a choice of Binary Concrete prior and an estimate of the KL divergence from that to the Binary Concrete approximate posterior [Maddison et al., 2017, Appendix C.3.2].

mators inadequate. For ST Gumbel-Softmax the situation is different since, after training, z is given discrete treatment throughout. Its success shows that, unlike in the other tasks considered, training on biased gradients is not too problematic.

5.6 Subsequent Work

Correia et al. [2020] proved to be a relevant contribution to the field of discrete and structured latent variable modeling. In particular, and most closely related to our work, Chen et al. [2021] added another method to the family of efficient marginalization techniques that we started in Correia et al. [2020]. They introduce a new function called *ev-softmax* that has its origins in evidential theory and evidential sparsification [Itkina et al., 2020]. The properties of this function are more similar to *softmax* than *sparsemax*, while still allowing for a sparse distribution. They show that their method is less prone to multimodality collapse [Itkina et al., 2020] and show improved results in the experiment described in §5.5.1.

In another application of sparsity for discrete latent variable models, Farinhas et al. [2022] introduced a new family of mixed discrete-continuous random variables that can be used to train these models. Their method has its origins in the recent field of sparse and continuous distributions [Martins et al., 2020]. Their method is applied to the experiments that we described in §5.5.2 and §5.5.3 and show promising results. As in our case, their method also bypasses the need for methods with high variance, as in SFE, and relaxations of the discrete space, as in Gumbel-Softmax; however, in their case, that's not due to explicit marginalization but rather due to the nature of the mixed random variables that they introduce.

As we will explore further in the next chapter, we believe that the methods we have proposed here have potential for structured applications. In particular, due to the structured nature of language, the explicit marginalization of the structured space can prove valuable to NLP. For example, our method has been applied successfully for semantic

parsing [Wang et al., 2021]. In this work, the authors use our top-k sparsemax method in the E-step of an EM algorithm. Since they do not have access to a true top-k, they approximate it using beam search. When compared to the other baselines and other methods proposed, the approach that used top-k sparsemax had the best performance.

5.7 Final Remarks and Chapter Summary

The solutions that were available to train discrete latent variable models greatly relied on MC sampling, which can have high variance. Methods that aim to decrease this variance are often not trivial to train and implement and may disincentivize practitioners from using this class of models. However, we believe that discrete latent variable models have, in many cases, more interpretable, intuitive, and compact latent representations.

We described a novel training strategy for discrete latent variable models, eschewing the common approach based on MC gradient estimation in favor of deterministic, exact marginalization under a sparse distribution. Our training strategy offers: a simple approach in implementation to training these models; no addition in the number of parameters; low increase in computational overhead (especially when compared to more sophisticated methods of variance reduction [Liu et al., 2019]); and improved performance. Sparsity leads to a powerful **adaptive** method, which can investigate fewer or more latent assignments z depending on the ambiguity of a training instance x, as well as on the stage in training.

We showcase the performance and flexibility of our methods by investigating a variety of applications, with both discrete and structured latent variables, with positive results. Our models very quickly approach a small number of latent assignment evaluations per sample, but make progress much faster and overall lead to superior results.

Our proposed method thus offers the accuracy and robustness of exact marginalization while meeting the efficiency and flexibility of sampling methods, providing a promising alternative to successfully train more **compact** neural models. As we are learning from

subsequent works (§5.6), we believe that our method can be used to address the challenges of training these sorts of models in a variety of applications, particularly NLP.

6

Conclusions

Contents

To wrap up the present thesis, we review the main contributions developed in this work, and discuss their impact, limitations, potential directions for future work, and their broader impact and ethical implications.

6.1 Summary of Contributions

Throughout this thesis, we have developed models and techniques that aim to make neural models more data-efficient, transparent, and compact.

In Chapter 3, we achieved **data-efficiency** by leveraging the **transfer learning** power of a pre-trained language model to avoid the need for producing a large synthetic dataset, in the context of a real-world application: Automatic Post-Editing.

In Chapter 4, we proposed a model with increased **transparency** by letting it **learn its own sparsity**. The chosen application was NMT, and we showed how the sparsity can help to interpret the various roles of each attention head in the model.

Finally, in Chapter 5, we once again leveraged **learnable sparsity** by introducing a new method to train latent variable models with discrete or structured nodes. We show the efficacy of our method in several applications: semi-supervised learning, emergent communication, and unsupervised learning of bit-vector representations. This method achieves better performance than standard methodologies for training discrete latent variable models and as such aids in the development of more **compact** neural models.

A common theme in this thesis is sparsity, particularly sparse mappings into the simplex. We have followed in the footsteps of work started by Martins and Astudillo [2016] and proposed novel uses of these sparse mappings and promising alternatives to them. In all works described, we have shown how in different applications we can let neural models learn how much sparsity they need; be it in the context of an attention mechanism, or in the context of the number of relevant latent assignments during training.

On another front, we tackled some forms of **weak supervision**. On one hand, we use transfer learning and minimize the amount of data required to train an APE model

in Chapter 3. On the other hand, we also applied our method of Chapter 5 to obtain improved performance on a semi-supervised learning task. Additionally, in §4.6 we saw how our discovered attention head roles can be used to fix attention patterns in a transformer, increasing inductive bias in the process [Raganato et al., 2020].

6.2 Open Problems and Limitations

In the pursuit of diligent research, we briefly discuss some of the open problems and limitations of the work presented in this thesis. We seek not to diminish any of these limitations, but rather present them fairly. We will also discuss the ethical ramifications of our work in a later section (§6.4).

As previously stated, sparsity is a prominent theme in this work. That sparsity is achieved through probability normalization functions such as sparsemax, α-entmax, and SparseMAP. While sparsity can lead to increased efficiency, we do not fully take advantage of this in the present thesis or in the publicly released code. For example, for α-entmax, further speed-ups are possible, with careful engineering, by leveraging more parallelism in the bisection algorithm for computing it. This is a relevant research direction, as it is now possible to exploit recent generations of GPU architectures which are more sparse-friendly than before.

In Chapter 3, we have used a large pre-trained model to circumvent the need to train MT models to produce a large synthetic dataset and then train an APE model on it. Although we believe this to be a way to spare resources, we do not wish to underplay the resources used to train the pre-trained model itself.

While we have shown promising results in Chapter 5, we are aware that the experiments we conducted were performed on toy datasets. While this may overemphasize the success of the work, we have seen in §5.6 that our method has already been successfully applied to applications that have real-world datasets. We hope to continue seeing this trend in the future.

6.3 Future Directions

We further list in this section exciting directions for future work that draw inspiration from the methods proposed in this thesis. We hope this might inspire other researchers in the field to explore these directions and extend the ideas presented.

Semi-supervised learning. Some forms of weak supervision were explored in this work. In particular, we have shown promising results in Chapter 5 by introducing a new method to train discrete latent variable models, which can be used to train semi-supervised models (§5.5.1). We believe that our method can be used in real-world applications where supervision is limited and that it will lead to better results than standard methods. For example, in NLP, there are many applications where a relaxation of the discrete space is not possible [*e.g.*, Lee et al., 2019]. We can consider the semi-supervised loss

$$\mathcal{L}_x(\theta) = \sum_{z \in \mathcal{Z}} \pi(z|x)\ell(x, z; \theta) + \sum_{z \in \mathcal{Z}} \pi(z|x, y)\ell(x, y, z; \theta) + \mathcal{R}(\theta)\,,$$

which is very similar to the loss used in §5.5.1, and apply it to more challenging datasets. Whereas before we could only rely on high variance methods if we wished to train such a model, we can now circumvent this limitation by using our training method and computing deterministic gradients.

Non-differentiability of the latent variable model learning signal. We note how, in Equation 2.17, the training of the parameters that learn $\pi(z|x; \theta)$ is decoupled from the training of the parameters of the generative model of the downstream loss: when marginalizing the latent variables, the downstream loss only scales the gradients computed from $\pi(z|x; \theta)$. This means that our method can still be applied when the downstream loss component does not have any trainable parameters. Furthermore, that component in Equation 2.17 can be replaced by any function, even non-differentiable, akin to a reward signal. One example of a real-world application where the downstream loss doesn't have learnable parameters is the prompting of large pre-trained language mod-

els [Liu et al., 2021]. In this line of work, the pre-trained model is assumed to already have the knowledge required to handle the task at hand stored in its parameters and only a prompt is required to generate the desired output. This problem could be framed as a latent variable model and one could interpret a natural language prompt as a discrete latent variable. Using the property we just described, the downstream task can be optimized via any differentiable or non-differentiable reward signal

$$\mathscr{L}_x(\theta) = -\sum_{z \in \mathcal{Z}} \pi(z|x;\theta) r(x,z),$$

where note that r does not depend on the model's parameters θ.

Latent draft translation. Related to the overall present thesis topics of increasing transparency and compactness of neural models in NLP, we highlight the potential of prototype editing [Guu et al., 2018, He et al., 2020]. In this generative model, the generative process has two main steps: (1) selecting a prototype sequence of tokens $\boldsymbol{t}$ from a list of possible prototypes (*e.g.*, the training data), and (2) editing that prototype by sampling a latent edit vector $\boldsymbol{z}$ that encodes the type of edit (*e.g.*, active to passive, reordering, *inter alia*). We note the promising possibility of using this approach to NMT: While He et al. [2020] apply their sparse neural editor to language modeling, we will model instead a conditional distribution over $\mathcal{X} \times \mathcal{Y}$:

$$p_{YTZ}(y, \boldsymbol{t}, \boldsymbol{z}|\boldsymbol{x}) = p_T(\boldsymbol{t}|\boldsymbol{x};\theta) p_Z(\boldsymbol{z}|\boldsymbol{x};\theta) p_{Y|TZ}(\boldsymbol{y}|\boldsymbol{t}, \boldsymbol{z};\theta)\,.$$

We may think of $\boldsymbol{t}$ as being a template for y — it can be a lexicalized rule with gaps or variables, a phrase-based mapping from a section of $\boldsymbol{x}$ to a string in the target language, among others. Once again, for deterministic gradients and greater training stability, we may use the training method proposed in Chapter 5.

6.4 Broader Impact

We discuss here the broader impact of our work.

First of all, we note that all of our code has been open-sourced to ensure it's scrutinizable by anyone and to boost any related future work that other researchers might want to pursue.

A common theme in this thesis is transparency. The main objective of improving such property in neural models is to have superior explanatory power and therefore to aid in understanding cases in which the model failed the downstream task. Interpretability of deep neural models can be essential to better discover any ethically harmful biases that exist in the data or in the model itself.

On the other hand, the models discussed in this work may also pave the way for malicious use cases, such as is the case for generative models with *Deepfakes*, fake human avatars used by malevolent Twitter users, or generation models that automatically generate fraudulent news, often based on the large pre-trained language models used in this thesis. Particularly, generative models as the ones used in Chapter 5 are remarkable at sampling new instances of fake data and, with the power of latent variables, the interpretability discussed before can be used maliciously to further push harmful biases instead of removing them. Furthermore, our work in Chapter 5 is promising in improving the performance of latent variable models with several discrete variables, that can be trained as attributes to control the sample generation. Attributes that can be activated or deactivated at will to generate fake data can both help beneficial and malignant users to finely control the generated sample. Our work may be currently agnostic to this, but we recognize the dangers and encourage effort to combating any malicious applications.

Energy-wise, our data-efficient and compact models aim to require less data and computation than other models that rely on a massive amount of data and infrastructure. This makes our models ideal for situations where data is scarce, or where there are few computational resources to train large models. We believe that data efficiency and compactness

are a step forward in the direction of alleviating environmental concerns of deep learning research [Strubell et al., 2019]. However, we note that, for example, the adaptive sparsity of the models proposed in Chapter 5 tend to use more resources earlier on than standard methods. Even though in the applications shown they consume much less as training progresses, it's not clear if that trend is still observed in all potential applications.

Regarding sparsity, we note how sparsity may exhibit a larger risk of disregarding certain correlations or groups of observations, and thus may contribute to misinforming a practitioner. Where such a model that uses sparsity informs decision-makers on matters that affect lives, these decisions may be based on an incomplete view of the correlations in the data and/or these correlations may be exaggerated in harmful ways. At this point, it is unclear to which extent this happens and, if it does, whether it is consistent across various uses. We believe that a better understanding of the impact of sparsity in potentially oversimplifying correlations is an additional interesting avenue of future work.

Bibliography

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Proceedings of NeurIPS*, 2017.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of ICLR*, 2015.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*, 2019.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Proceedings of NeurIPS*, 2020.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of ACL*, 2019.

Marcin Junczys-Dowmunt, Kenneth Heafield, Hieu Hoang, Roman Grundkiewicz, and Anthony Aue. Marian: Cost-effective High-Quality Neural Machine Translation in C++. In *Proceedings of WNMT*, 2018a.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling Neural Machine Translation. In *Proceedings of WMT*, 2018.

Yoon Kim, Sam Wiseman, and Alexander M. Rush. A Tutorial on Deep Latent Variable Models of Natural Language. *preprint arXiv:1812.06834*, 2018.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *Proceedings of ICLR*, 2014.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of ICML*, 2014.

Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised Learning with Deep Generative Models. In *Proceedings of NeurIPS*, 2014.

Ivan Titov and Ryan McDonald. A Joint Model of Text and Aspect Ratings for Sentiment Summarization. In *Proceedings of ACL*, 2008.

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. Interpretable Neural Predictions with Differentiable Binary Variables. In *Proceedings of ACL*, 2019.

Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4):229–256, 1992.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of ICLR*, 2017.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: a Continous Relaxation of Discrete Random Variables. In *Proceedings of ICLR*, 2017.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of NAACL*, 2018.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*, 2015.

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. *JMLR*, 21(140):1–67, 2020.

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 ACL*, 2020.

André F.T. Martins and Ramón Fernandez Astudillo. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In *Proceedings of ICML*, 2016.

Vlad Niculae and Mathieu Blondel. A Regularized Framework for Sparse and Structured Neural Attention. In *Proceedings of NeurIPS*, 2017.

Ben Peters, Vlad Niculae, and André F.T. Martins. Sparse Sequence-to-Sequence Models. In *Proceedings of ACL*, 2019.

Sameen Maruf, André F.T. Martins, and Gholamreza Haffari. Selective Attention for Context-aware Neural Machine Translation. In *Proceedings of NAACL-HLT*, 2019.

Chaitanya Malaviya, Pedro Ferreira, and André F.T. Martins. Sparse and Constrained Attention for Neural Machine Translation. In *Proceedings of ACL*, 2018.

Vlad Niculae, André F.T. Martins, Mathieu Blondel, and Claire Cardie. SparseMAP: Differentiable Sparse Structured Inference. In *Proceedings of ICML*, 2018a.

Vlad Niculae, André F.T. Martins, and Claire Cardie. Towards Dynamic Computation Graphs via Sparse Latent Structure. In *Proceedings of EMNLP*, 2018b.

Marcos Treviso, António Góis, Patrick Fernandes, Erick Fonseca, and André F. T. Martins. Predicting Attention Sparsity in Transformers. In *Proceedings of SPNLP*, 2022.

António Farinhas, Wilker Aziz, Vlad Niculae, and André F. T. Martins. Sparse Communication via Mixed Distributions. In *Proceedings of ICLR*, 2022.

Gonçalo M. Correia and André F. T. Martins. A Simple and Effective Approach to Automatic Post-Editing with Transfer Learning. In *Proceedings of ACL*, 2019.

António V. Lopes, M. Amin Farajian, Gonçalo M. Correia, Jonay Trenous, and André F. T. Martins. Unbabel's Submission to the WMT2019 APE Shared Task: BERT-based Encoder-Decoder for Automatic Post-Editing. In *Proceedings of WMT*, 2019.

Gonçalo M Correia, Vlad Niculae, and André F.T. Martins. Adaptively Sparse Transformers. In *Proceedings of EMNLP*, 2019.

Gonçalo M. Correia, Vlad Niculae, Wilker Aziz, and André F. T. Martins. Efficient Marginalization of Discrete and Structured Latent Variables via Sparsity. In *Proceedings of NeurIPS*, 2020.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of ICLR*, 2015.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of EMNLP*, 2015.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional Sequence to Sequence Learning. In *Proceedings of ICML*, 2017.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. *preprint*, 2018.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A Primer in BERTology: What We Know About How BERT Works. *TACL*, 8:842–866, 2020.

Sarthak Jain and Byron C. Wallace. Attention is not Explanation. In *Proceedings of NAACL-HLT*, 2019.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning Sparse Neural Networks through L_0 Regularization. In *Proceedings of ICLR*, 2018.

Wenqi Shao, Tianjian Meng, Jingyu Li, Ruimao Zhang, Yudian Li, Xiaogang Wang, and Ping Luo. SSN: Learning Sparse Switchable Normalization via SparsestMax. In *Proceedings of CVPR*, 2019.

Mathieu Blondel, André F.T. Martins, and Vlad Niculae. Learning Classifiers with Fenchel-Young Losses: Generalized Entropies, Margins, and Algorithms. In *Proceedings of AISTATS*, 2019.

Michael Held, Philip Wolfe, and Harlan P. Crowder. Validation of Subgradient Optimization. *Mathematical Programming*, 6(1):62–88, 1974.

Laurent Condat. Fast Projection onto the Simplex and the ℓ_1 Ball. *Mathematical Programming*, 158(1-2):575–585, 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of NeurIPS*, 2019.

Constantino Tsallis. Possible Generalization of Boltzmann-Gibbs Statistics. *Journal of Statistical Physics*, 52:479–487, 1988.

Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer New York, 1999.

André F.T. Martins, Mário A.T. Figueiredo, Pedro M.Q. Aguiar, Noah A. Smith, and Eric P. Xing. AD3: Alternating Directions Dual Decomposition for MAP Inference in Graphical Models. *JMLR*, 16(1):495–545, 2015.

Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

Reuven Y. Rubinstein. A Monte Carlo Method for Estimating the Gradient in a Stochastic Network. *Unpublished manuscript, Technion, Haifa, Israel*, 1976.

John Paisley, David M. Blei, and Michael I. Jordan. Variational Bayesian Inference with Stochastic Search. In *Proceedings of ICML*, 2012.

Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih. MuProp: Unbiased Backpropagation for Stochastic Neural Networks. In *Proceedings of ICLR*, 2016.

Chong Wang, Xi Chen, Alexander J. Smola, and Eric P. Xing. Variance Reduction for Stochastic Gradient Optimization. In *Proceedings of NeurIPS*, 2013.

George Tucker, Andriy Mnih, Chris J. Maddison, John Lawson, and Jascha Sohl-Dickstein. REBAR: Low-variance, Unbiased Gradient Estimates for Discrete Latent Variable Models. In *Proceedings of NeurIPS*, 2017.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the Void: Optimizing Control Variates for Black-box Gradient Estimation. In *Proceedings of ICLR*, 2018.

George Casella and Christian P Robert. Rao-Blackwellisation of Sampling Schemes. *Biometrika*, 83(1):81–94, 1996.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In *Proceedings of AISTATS*, 2014.

Runjing Liu, Jeffrey Regier, Nilesh Tripuraneni, Michael Jordan, and Jon Mcauliffe. Rao-Blackwellized Stochastic Gradients for Discrete Distributions. In *Proceedings of ICML*, 2019.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *preprint arXiv:1308.3432*, 2013.

Martin J. Wainwright and Michael I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

Michel Simard, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. Rule-Based Translation with Statistical Phrase-Based Post-Editing. In *Proceedings of WMT*, 2007.

Vicent Alabau, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Ulrich Germann, Jesús González-Rubio, Robin Hill, Philipp Koehn, Luis Leiva, Bartolomé Mesa-Lao, Daniel Ortiz-Martínez, Herve Saint-Amand, Germán Sanchis Trilles, and Chara Tsoukala. CASMACAT: A Computer-assisted Translation Workbench. In *Proceedings of the Demonstrations at EACL*, 2014.

Marcello Federico, Nicola Bertoldi, Mauro Cettolo, Matteo Negri, Marco Turchi, Marco Trombetti, Alessandro Cattelan, Antonio Farina, Domenico Lupinetti, Andrea Martines, Alberto Massidda, Holger Schwenk, Loïc Barrault, Frederic Blain, Philipp Koehn, Christian Buck, and Ulrich Germann. The MateCat Tool. In *Proceedings of COLING*, 2014.

Michael Denkowski. *Machine Translation for Human Translators*. PhD thesis, Carnegie Mellon University, 2015.

Christopher M. Hokamp. *Deep Interactive Text Prediction and Quality Estimation in Translation Interfaces*. PhD thesis, Dublin City University, 2018.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. Log-linear Combinations of Monolingual and Bilingual Neural Machine Translation Models for Automatic Post-Editing. In *Proceedings of WMT*, 2016.

Matteo Negri, Marco Turchi, Rajen Chatterjee, and Nicola Bertoldi. eSCAPE: a Large-scale Synthetic Corpus for Automatic Post-Editing. In *Proceedings of LREC*, 2018.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. MS-UEdin Submission to the WMT2018 APE Shared Task: Dual-Source Transformer for Automatic Post-Editing. In *Proceedings of WMT*, 2018.

Amirhossein Tebbifakhr, Ruchit Agrawal, Matteo Negri, and Marco Turchi. Multi-source Transformer with Combined Losses for Automatic Post-Editing. In *Proceedings of WMT*, 2018.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in Translation: Contextualized Word Vectors. In *Proceedings of NeurIPS*, 2017.

Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL*, 2018.

Devendra Sachan and Graham Neubig. Parameter Sharing Methods for Multilingual Self-Attentional Translation Models. In *Proceedings of WMT*, 2018.

Alexandre Bérard, Laurent Besacier, and Olivier Pietquin. LIG-CRIStAL Submission for the WMT 2017 Automatic Post-Editing Task. In *Proceedings of WMT*, 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 BlackboxNLP*, 2018.

Guillaume Lample and Alexis Conneau. Cross-lingual Language Model Pretraining. In *Proceedings of NeurIPS*, 2019.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proceedings of ACL*, 2017.

Rajen Chatterjee, Matteo Negri, Raphael Rubino, and Marco Turchi. Findings of the WMT 2018 Shared Task on Automatic Post-Editing. In *Proceedings of WMT*, 2018.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *preprint arXiv:1609.08144*, 2016.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of AMTA*, 2006.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*, 2002.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *JMLR*, 15(1):1929–1958, 2014.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing Neural Networks by Penalizing Confident Output Distributions. In *Proceedings of ICLR*, 2017.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. Marian: Fast Neural Machine Translation in C++. In *Proceedings of ACL*, 2018b.

Haoyu Zhang, Jianjun Xu, and Ji Wang. Pretraining-Based Natural Language Generation for Text Summarization. In *Proceedings of CoNLL*, 2019.

Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. Distilling Knowledge Learned in BERT for Text Generation. In *Proceedings of ACL*, 2020.

Jihyung Lee, WonKee Lee, Jaehun Shin, Baikjin Jung, Young-Kil Kim, and Jong-Hyeok Lee. POSTECH-ETRI's Submission to the WMT2020 APE Shared Task: Automatic Post-Editing with Cross-lingual Language Model. In *Proceedings of WMT*, 2020.

Jiayi Wang, Ke Wang, Kai Fan, Yuqi Zhang, Jun Lu, Xin Ge, Yangbin Shi, and Yu Zhao. Alibaba's Submission for the WMT 2020 APE Shared Task: Improving Automatic Post-Editing with Pre-trained Conditional Cross-Lingual BERT. In *Proceedings of WMT*, 2020.

António Góis, Kyunghyun Cho, and André Martins. Learning Non-Monotonic Automatic Post-Editing of Translations from Human Orderings. In *Proceedings of EAMT*, 2020.

Shamil Chollampatt, Raymond Hendy Susanto, Liling Tan, and Ewa Szymanska. Can Automatic Post-Editing Improve NMT? In *Proceedings of EMNLP*, 2020.

Takashi Kodama, Ryuichiro Higashinaka, Koh Mitsuda, Ryo Masumura, Yushi Aono, Ryuta Nakamura, Noritake Adachi, and Hidetoshi Kawabata. Generating Responses That Reflect Meta Information in User-Generated Question Answer Pairs. In *Proceedings of LREC*, 2020.

Xuancheng Huang, Jingfang Xu, Maosong Sun, and Yang Liu. Transfer Learning for Sequence Generation: From Single-source to Multi-source. In *Proceedings of ACL*, 2021.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. Why Self-attention? A Targeted Evaluation of Neural Machine Translation Architectures. In *Proceedings of EMNLP*, 2018.

Alessandro Raganato and Jörg Tiedemann. An Analysis of Encoder Representations in Transformer-based Machine Translation. In *Proceedings of BlackboxNLP*, 2018.

David Mareček and Rudolf Rosa. Extracting Syntactic Trees from Transformer Encoder Self-attentions. In *Proceedings of BlackboxNLP*, 2018.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of ACL*, 2019.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of ACL*, 2019.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. Latent Alignment and Variational Attention. In *Proceedings of NeurIPS*, 2018.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers. *preprint arXiv:1904.10509*, 2019.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive Attention Span in Transformers. In *Proceedings of ACL*, 2019.

Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. Pay Less Attention with Lightweight and Dynamic Convolutions. In *Proceedings of ICLR*, 2019.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. Multi-Head Attention with Disagreement Regularization. In *Proceedings of EMNLP*, 2018.

Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On Differentiating Parameterized Argmin and Argmax Problems with Application to Bi-level Optimization. *preprint arXiv:1607.05447*, 2016.

Brandon Amos and J. Zico Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *Proceedings of ICML*, 2017.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Niehues Jan, Stüker Sebastian, Sudoh Katsuitho, Yoshino Koichiro, and Federmann Christian. Overview of the IWSLT 2017 Evaluation Campaign. In *Proceedings of IWSLT*, 2017.

Graham Neubig. The Kyoto free translation task. `https://www.phontron.com/kftt`, 2011.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 Conference on Machine Translation. In *Proceedings of WMT*, 2016.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of WMT*, 2014.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of ACL*, 2016.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. Context-aware Neural Machine Translation Learns Anaphora Resolution. In *Proceedings of ACL*, 2018.

Giannis Daras, Nikita Kitaev, Augustus Odena, and Alexandros G. Dimakis. SMYRF: Efficient Attention Using Asymmetric Clustering. In *Proceedings of NeurIPS*, 2020.

Xiaoya Li, Yuxian Meng, Mingxin Zhou, Qinghong Han, Fei Wu, and Jiwei Li. SAC: Accelerating and Structuring Self-Attention via Sparse Adaptive Connection. In *Proceedings of NeurIPS*, 2020.

William Merrill, Vivek Ramanujan, Yoav Goldberg, Roy Schwartz, and Noah Smith. Effects of Parameter Norm Growth During Transformer Training: Inductive Bias from Gradient Descent. In *Proceedings of EMNLP*, 2021.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient Content-Based Sparse Attention with Routing Transformers. *TACL*, 9:53–68, 2021.

Jyun-Yu Jiang, Chenyan Xiong, Chia-Jung Lee, and Wei Wang. Long Document Ranking with Query-Directed Sparse Transformer. In *Proceedings of EMNLP*, 2020.

Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise Self-Attention for Long Document Understanding. In *Proceedings of EMNLP*, 2020.

Sainbayar Sukhbaatar, Da Ju, Spencer Poff, Stephen Roller, Arthur Szlam, Jason Weston, and Angela Fan. Not All Memories Are Created Equal: Learning to Forget by Expiring. In *Proceedings of ICML*, 2021.

Weiqiu You, Simeng Sun, and Mohit Iyyer. Hard-Coded Gaussian Attention for Neural Machine Translation. In *Proceedings of ACL*, 2020.

Madhura Pande, Aakriti Budhraja, Preksha Nema, Pratyush Kumar, and Mitesh M. Khapra. The Heads Hypothesis: A Unifying Statistical Approach towards Understanding Multi-Headed Attention in BERT. In *Proceedings of AAAI*, 2021.

Zhijiang Guo, Guoshun Nan, Wei Lu, and Shay B Cohen. Learning Latent Forests for Medical Relation Extraction. In *Proceedings of IJCAI*, 2020.

Boxiang Yun, Yan Wang, Jieneng Chen, Huiyu Wang, Wei Shen, and Qingli Li. SpecTr: Spectral Transformer for Hyperspectral Pathology Image Segmentation. *preprint arXiv:2103.03604*, 2021.

Tianchu Ji, Shraddhan Jain, Michael Ferdman, Peter Milder, H. Andrew Schwartz, and Niranjan Balasubramanian. On the Distribution, Sparsity, and Inference-time Quantization of Attention Values in Transformers. In *Proceedings of ACL-IJCNLP*, 2021.

Hongfei Xu, Qiuhui Liu, Josef van Genabith, and Deyi Xiong. Learning Hard Retrieval Decoder Attention for Transformers. In *Proceedings of EMNLP*, 2021.

Chulhee Yun, Yin-Wen Chang, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. $\mathcal{O}(n)$ Connections Are Expressive Enough: Universal Approximability of Sparse Transformers. In *Proceedings of NeurIPS*, 2020.

Biao Zhang, Ivan Titov, and Rico Sennrich. Sparse Attention with Linear Units. In *Proceedings of EMNLP*, 2021a.

Biao Zhang, Ivan Titov, and Rico Sennrich. On Sparsifying Encoder Outputs in Sequence-to-Sequence Models. In *Proceedings of ACL-IJCNLP*, 2021b.

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. Fixed Encoder Self-Attention Patterns in Transformer-Based Machine Translation. In *Proceedings of EMNLP*, 2020.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte Carlo Gradient Estimation in Machine Learning. *preprint arXiv:1906.10652*, 2019.

Kris Cao. *Learning Meaning Representations for Text Generation with Deep Generative Models*. PhD thesis, University of Cambridge, 2019.

Andriy Mnih and Danilo J. Rezende. Variational Inference for Monte Carlo Objectives. In *Proceedings of ICML*, 2016.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance Weighted Autoencoders. In *Proceedings of ICLR*, 2016.

Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning. *JMLR*, 5:1471–1530, 2004.

Guy Lorberbom, Andreea Gane, Tommi Jaakkola, and Tamir Hazan. Direct Optimization through arg max for Discrete Variational Auto-Encoder. In *Proceedings of NeurIPS*, 2019.

Marin Vlastelica, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of Blackbox Combinatorial Solvers. In *Proceedings of ICLR*, 2020.

Mingzhang Yin and Mingyuan Zhou. ARM: Augment-REINFORCE-Merge Gradient for Stochastic Binary Networks. In *Proceedings of ICLR*, 2019.

Art B. Owen. *Monte Carlo Theory, Methods and Examples*. Copyright Art Owen, 2013.

Wouter Kool, Herke van Hoof, and Max Welling. Estimating Gradients for Discrete Random Variables by Sampling without Replacement. In *Proceedings of ICLR*, 2020.

Anastasios Kyrillidis, Stephen Becker, Volkan Cevher, and Christoph Koch. Sparse Projections onto the Simplex. In *Proceedings of ICML*, 2013.

Venkata Krishna Pillutla, Vincent Roulet, Sham M. Kakade, and Zaid Harchaoui. A Smoother Way to Train Structured Prediction Models. In *Proceedings of NeurIPS*, 2018.

Mathieu Blondel, André F.T. Martins, and Vlad Niculae. Learning with Fenchel-Young losses. *JMLR*, 21(35):1–69, 2020.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent Cooperation and the Emergence of (Natural) Language. In *Proceedings of ICLR*, 2017.

Eugene Kharitonov, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni. EGG: a Toolkit for Research on Emergence of LanGuage in Games. In *Proceedings of EMNLP*, 2019.

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.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical Sequence Training for Image Captioning. In *Proceedings of CVPR*, 2017.

Andriy Mnih and Karol Gregor. Neural Variational Inference and Learning in Belief Networks. In *Proceedings of ICML*, 2014.

Simon Kirby. Natural Language from Artificial Life. *Artificial life*, 8(2):185–215, 2002.

Serhii Havrylov and Ivan Titov. Emergence of Language with Multi-agent Games: Learning to Communicate with Sequences of Symbols. In *Proceedings of NeurIPS*, 2017.

Emilio Jorge, Mikael Kågebäck, Fredrik D. Johansson, and Emil Gustavsson. Learning to Play Guess Who? and Inventing a Grounded Language as a Consequence. In *Proceedings of NeurIPS Workshop on Deep Reinforcement Learning*, 2016.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to Communicate with Deep Multi-agent Reinforcement Learning. In *Proceedings of NeurIPS*, 2016.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning Multiagent Communication with Backpropagation. In *Proceedings of NeurIPS*, 2016.

Volodymyr Mnih, Adria Puigdomenech Badia, Lehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of ICML*, 2016.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *preprint arXiv:1708.07747*, 2017.

Alexander A. Alemi, Ben Poole, Ian Fischer, Joshua V. Dillon, Rif A. Saurous, and Kevin Murphy. Fixing a Broken ELBO. In *Proceedings of ICML*, 2018.

Phil Chen, Masha Itkina, Ransalu Senanayake, and Mykel J. Kochenderfer. Evidential Softmax for Sparse Multimodal Distributions in Deep Generative Models. In *Proceedings of NeurIPS*, 2021.

Masha Itkina, Boris Ivanovic, Ransalu Senanayake, Mykel J. Kochenderfer, and Marco Pavone. Evidential Sparsification of Multimodal Latent Spaces in Conditional Variational Autoencoders. In *Proceedings of NeurIPS*, 2020.

André F. T. Martins, António Farinhas, Marcos Treviso, Vlad Niculae, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. Sparse and Continuous Attention Mechanisms. In *Proceedings of NeurIPS*, 2020.

Bailin Wang, Mirella Lapata, and Ivan Titov. Learning from Executions for Semantic Parsing. In *Proceedings of NAACL*, 2021.

Mina Lee, Tatsunori B. Hashimoto, and Percy Liang. Learning Autocomplete Systems as a Communication Game. In *Proceedings of the NeurIPS Workshop on Emergent Communication*, 2019.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *preprint arXiv:2107.13586*, 2021.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. Generating Sentences by Editing Prototypes. *TACL*, 6:437–450, 2018.

Junxian He, Taylor Berg-Kirkpatrick, and Graham Neubig. Learning Sparse Prototypes for Text Generation. In *Proceedings of NeurIPS*, 2020.

Frank H Clarke. *Optimization and Nonsmooth Analysis*. SIAM, 1990.

A

Proof of Proposition 4.3

Jacobian of α-entmax *w.r.t.* the shape parameter α

Recall that the entmax transformation is defined as:

$$\alpha\text{-entmax}(\boldsymbol{z}) := \operatorname*{argmax}_{\boldsymbol{p}\in\triangle^{d-1}} \boldsymbol{p}^\top \boldsymbol{z} + \mathsf{H}^{\mathrm{T}}_\alpha(\boldsymbol{p}), \tag{A.1}$$

where $\alpha \geq 1$ and $\mathsf{H}^{\mathrm{T}}_\alpha$ is the Tsallis entropy,

$$\mathsf{H}^{\mathrm{T}}_\alpha(\boldsymbol{p}) := \begin{cases} \frac{1}{\alpha(\alpha-1)} \sum_j \left(p_j - p_j^\alpha\right), & \alpha \neq 1, \\ \mathbb{H}(\boldsymbol{p}), & \alpha = 1, \end{cases}$$

and $\mathbb{H}(\boldsymbol{p}) := -\sum_j p_j \log p_j$ is the Shannon entropy.

In this section, we derive the Jacobian of entmax with respect to the scalar parameter α.

General case of $\alpha > 1$

From the KKT conditions associated with the optimization problem in Eq. A.1, we have that the solution $\boldsymbol{p}^\star$ has the following form, coordinate-wise:

$$p_i^\star = [(\alpha - 1)(z_i - \tau^\star)]_+^{1/(\alpha-1)}, \tag{A.2}$$

where $\tau^\star$ is a scalar Lagrange multiplier that ensures that $\boldsymbol{p}^\star$ normalizes to 1, *i.e.*, it is defined implicitly by the condition:

$$\sum_i [(\alpha - 1)(z_i - \tau^\star)]_+^{1/(\alpha-1)} = 1. \tag{A.3}$$

For general values of α, Eq. A.3 lacks a closed form solution. This makes the computation of the Jacobian

$$\frac{\partial\, \alpha\text{-entmax}(\boldsymbol{z})}{\partial \alpha}$$

non-trivial. Fortunately, we can use the technique of implicit differentiation to obtain this Jacobian.

The Jacobian exists almost everywhere, and the expressions we derive expressions yield a generalized Jacobian [Clarke, 1990] at any non-differentiable points that may occur for certain $(\alpha, \boldsymbol{z})$ pairs. We begin by noting that $\frac{\partial p_i^\star}{\partial \alpha} = 0$ if $p_i^\star = 0$, because increasing α keeps sparse coordinates sparse.[1] Therefore we need to worry only about coordinates that are in the support of $\boldsymbol{p}^\star$. We will assume hereafter that the i^{th} coordinate of $\boldsymbol{p}^\star$ is non-zero. We have:

$$\begin{aligned}
\frac{\partial p_i^\star}{\partial \alpha} &= \frac{\partial}{\partial \alpha}[(\alpha-1)(z_i-\tau^\star)]^{\frac{1}{\alpha-1}} \\
&= \frac{\partial}{\partial \alpha}\exp\left[\frac{1}{\alpha-1}\log[(\alpha-1)(z_i-\tau^\star)]\right] \\
&= p_i^\star \frac{\partial}{\partial \alpha}\left[\frac{1}{\alpha-1}\log[(\alpha-1)(z_i-\tau^\star)]\right] \\
&= \frac{p_i^\star}{(\alpha-1)^2}\left[\frac{\frac{\partial}{\partial \alpha}[(\alpha-1)(z_i-\tau^\star)]}{z_i-\tau^\star} - \log[(\alpha-1)(z_i-\tau^\star)]\right] \\
&= \frac{p_i^\star}{(\alpha-1)^2}\left[\frac{z_i-\tau^\star-(\alpha-1)\frac{\partial \tau^\star}{\partial \alpha}}{z_i-\tau^\star} - \log[(\alpha-1)(z_i-\tau^\star)]\right] \\
&= \frac{p_i^\star}{(\alpha-1)^2}\left[1 - \frac{\alpha-1}{z_i-\tau^\star}\frac{\partial \tau^\star}{\partial \alpha} - \log[(\alpha-1)(z_i-\tau^\star)]\right]. \qquad \text{(A.4)}
\end{aligned}$$

We can see that this Jacobian depends on $\frac{\partial \tau^\star}{\partial \alpha}$, which we now compute using implicit differentiation.

Let $\mathcal{S} = \{i : p_i^\star > 0\}$). By differentiating both sides of Eq. A.3, re-using some of the

[1] This follows from the margin property of $\mathsf{H}_\alpha^{\mathrm{T}}$ [Blondel et al., 2019].

steps in Eq. A.4, and recalling Eq. A.2, we get

$$
\begin{aligned}
0 &= \sum_{i\in\mathcal{S}} \frac{\partial}{\partial\alpha}[(\alpha-1)(z_i-\tau^\star)]^{1/(\alpha-1)} \\
&= \sum_{i\in\mathcal{S}} \frac{p_i^\star}{(\alpha-1)^2}\left[1-\frac{\alpha-1}{z_i-\tau^\star}\frac{\partial\tau^\star}{\partial\alpha}-\log[(\alpha-1)(z_i-\tau^\star)]\right] \\
&= \frac{1}{(\alpha-1)^2}-\frac{\partial\tau^\star}{\partial\alpha}\sum_{i\in\mathcal{S}}\frac{p_i^\star}{(\alpha-1)(z_i-\tau^\star)}-\sum_{i\in\mathcal{S}}\frac{p_i^\star}{(\alpha-1)^2}\log[(\alpha-1)(z_i-\tau^\star)] \\
&= \frac{1}{(\alpha-1)^2}-\frac{\partial\tau^\star}{\partial\alpha}\sum_i (p_i^\star)^{2-\alpha}-\sum_i\frac{p_i^\star}{\alpha-1}\log p_i^\star \\
&= \frac{1}{(\alpha-1)^2}-\frac{\partial\tau^\star}{\partial\alpha}\sum_i (p_i^\star)^{2-\alpha}+\frac{\mathbb{H}(\boldsymbol{p}^*)}{\alpha-1},
\end{aligned}
\tag{A.5}
$$

from which we obtain:

$$
\frac{\partial\tau^\star}{\partial\alpha} = \frac{\frac{1}{(\alpha-1)^2}+\frac{\mathbb{H}(\boldsymbol{p}^\star)}{\alpha-1}}{\sum_i (p_i^\star)^{2-\alpha}}. \tag{A.6}
$$

Finally, plugging Eq. A.6 into Eq. A.4, we get:

$$
\begin{aligned}
\frac{\partial p_i^\star}{\partial\alpha} &= \frac{p_i^\star}{(\alpha-1)^2}\left[1-\frac{1}{(p_i^\star)^{\alpha-1}}\frac{\partial\tau^\star}{\partial\alpha}-(\alpha-1)\log p_i^\star\right] \\
&= \frac{p_i^\star}{(\alpha-1)^2}\left[1-\frac{1}{(p_i^\star)^{\alpha-1}}\frac{\frac{1}{(\alpha-1)^2}+\frac{\mathbb{H}(\boldsymbol{p}^\star)}{\alpha-1}}{\sum_i (p_i^\star)^{2-\alpha}}-(\alpha-1)\log p_i^\star\right] \\
&= \frac{p_i^\star-\tilde{p}_i(\alpha)}{(\alpha-1)^2}-\frac{p_i^\star\log p_i^\star+\tilde{p}_i(\alpha)\mathbb{H}(\boldsymbol{p}^\star)}{\alpha-1},
\end{aligned}
\tag{A.7}
$$

where we denote by

$$
\tilde{p}_i(\alpha) = \frac{(p_i^\star)^{2-\alpha}}{\sum_j (p_j^\star)^{2-\alpha}}.
$$

The distribution $\tilde{\boldsymbol{p}}(\alpha)$ can be interpreted as a "skewed" distribution obtained from $\boldsymbol{p}^\star$, which appears in the Jacobian of α-entmax($\boldsymbol{z}$) *w.r.t.* $\boldsymbol{z}$ as well Peters et al. [2019].

Solving the indetermination for $\alpha = 1$

We can write Eq. A.7 as

$$\frac{\partial p_i^\star}{\partial \alpha} = \frac{p_i^\star - \tilde{p}_i(\alpha) - (\alpha - 1)(p_i^\star \log p_i^\star + \tilde{p}_i(\alpha)\mathbb{H}(\boldsymbol{p}^\star))}{(\alpha - 1)^2}. \tag{A.8}$$

When $\alpha \to 1^+$, we have $\tilde{\boldsymbol{p}}(\alpha) \to \boldsymbol{p}^\star$, which leads to a $\frac{0}{0}$ indetermination.

To solve this indetermination, we will need to apply L'Hôpital's rule twice. Let us first compute the derivative of $\tilde{p}_i(\alpha)$ with respect to α. We have

$$\frac{\partial}{\partial \alpha}(p_i^\star)^{2-\alpha} = -(p_i^\star)^{2-\alpha} \log p_i^\star,$$

therefore

$$\begin{aligned}
\frac{\partial}{\partial \alpha}\tilde{p}_i(\alpha) &= \frac{\partial}{\partial \alpha}\frac{(p_i^\star)^{2-\alpha}}{\sum_j (p_j^\star)^{2-\alpha}} \\
&= \frac{-(p_i^\star)^{2-\alpha}\log p_i^\star \sum_j (p_j^\star)^{2-\alpha} + (p_i^\star)^{2-\alpha}\sum_j (p_j^\star)^{2-\alpha}\log p_j^\star}{\left(\sum_j (p_j^\star)^{2-\alpha}\right)^2} \\
&= -\tilde{p}_i(\alpha)\log p_i^\star + \tilde{p}_i(\alpha)\sum_j \tilde{p}_j(\alpha)\log p_j^\star.
\end{aligned} \tag{A.9}$$

Differentiating the numerator and denominator in Eq. A.8, we get:

$$\begin{aligned}
\frac{\partial p_i^\star}{\partial \alpha} &= \lim_{\alpha \to 1^+} \frac{(1 + (\alpha - 1)\mathbb{H}(\boldsymbol{p}^\star))\tilde{p}_i(\alpha)(\log p_i^\star - \sum_j \tilde{p}_j(\alpha)\log p_j^\star) - p_i^\star \log p_i^\star - \tilde{p}_i(\alpha)\mathbb{H}(\boldsymbol{p}^\star)}{2(\alpha - 1)} \\
&= A + B,
\end{aligned} \tag{A.10}$$

with

$$\begin{aligned}
A &= \lim_{\alpha \to 1^+} \frac{\mathbb{H}(\boldsymbol{p}^\star)\tilde{p}_i(\alpha)(\log p_i^\star - \sum_j \tilde{p}_j(\alpha)\log p_j^\star)\mathbb{H}(\boldsymbol{p}^\star)}{2} \\
&= \frac{\mathbb{H}(\boldsymbol{p}^\star)p_i^\star \log p_i^\star + p_i^\star(\mathbb{H}(\boldsymbol{p}^\star))^2}{2},
\end{aligned} \tag{A.11}$$

and

$$B = \lim_{\alpha \to 1^+} \frac{\tilde{p}_i(\alpha)(\log p_i^\star - \sum_j \tilde{p}_j(\alpha) \log p_j^\star) - p_i^\star \log p_i^\star - \tilde{p}_i(\alpha)\mathbb{H}(\boldsymbol{p}^\star)}{2(\alpha - 1)}. \tag{A.12}$$

When $\alpha \to 1^+$, B becomes again a $\frac{0}{0}$ indetermination, which we can solve by applying again L'Hôpital's rule. Differentiating the numerator and denominator in Eq. A.12:

$$\begin{aligned} B &= \frac{1}{2} \lim_{\alpha \to 1^+} \left\{ \tilde{p}_i(\alpha) \log p_i^\star \left(\sum_j \tilde{p}_j(\alpha) \log p_j^\star - \log p_i^\star \right) \right. \\ &\quad -\tilde{p}_i(\alpha) \left(\sum_j \tilde{p}_j(\alpha) \log p_j^\star - \log p_i^\star \right) \left(\sum_j \tilde{p}_j(\alpha) \log p_j^\star + \mathbb{H}(\boldsymbol{p}^\star) \right) \\ &\quad \left. -\tilde{p}_i(\alpha) \sum_j \tilde{p}_j(\alpha) \log p_j^\star \left(\sum_k \tilde{p}_k(\alpha) \log p_k^\star - \log p_j^\star \right) \right\} \\ &= \frac{-p_i^\star \log p_i^\star (\mathbb{H}(\boldsymbol{p}^\star) + \log p_i^\star) + p_i^\star \sum_j p_j^\star \log p_j^\star (\mathbb{H}(\boldsymbol{p}^\star) + \log p_j^\star)}{2} \\ &= \frac{-\mathbb{H}(\boldsymbol{p}^\star) p_i^\star \log p_i^\star - p_i^\star (\mathbb{H}(\boldsymbol{p}^\star))^2 - p_i^\star \log^2 p_i^\star + p_i^\star \sum_j p_j^\star \log^2 p_j^\star}{2}. \end{aligned} \tag{A.13}$$

Finally, summing Eq. A.11 and Eq. A.13, we get

$$\left. \frac{\partial p_i^\star}{\partial \alpha} \right|_{\alpha=1} = \frac{-p_i^\star \log^2 p_i^\star + p_i^\star \sum_j p_j^\star \log^2 p_j^\star}{2}. \tag{A.14}$$

Summary

To sum up, we have the following expression for the Jacobian of α-entmax with respect to α:

$$\frac{\partial p_i^\star}{\partial \alpha} = \begin{cases} \frac{p_i^\star - \tilde{p}_i(\alpha)}{(\alpha-1)^2} - \frac{p_i^\star \log p_i^\star + \tilde{p}_i(\alpha)\mathbb{H}(\boldsymbol{p}^\star)}{\alpha - 1}, & \text{for } \alpha > 1 \\ \frac{-p_i^\star \log^2 p_i^\star + p_i^\star \sum_j p_j^\star \log^2 p_j^\star}{2}, & \text{for } \alpha = 1. \end{cases}$$

B

Infrastructure

The infrastructure used during the course of this project consisted of 4 machines with the specifications shown in Table B.1. The machines were used interchangeably, and all experiments were executed in a single GPU. Despite having machines with different specifications, we did not observe large differences in the execution time of our models across different machines.

#	GPU	CPU
1.	4 × Titan Xp - 12GB	16 × AMD Ryzen 1950X @ 3.40GHz - 128GB
2.	4 × GTX 1080 Ti - 12GB	8 × Intel i7-9800X @ 3.80GHz - 128GB
3.	3 × RTX 2080 Ti - 12GB	12 × AMD Ryzen 2920X @ 3.50GHz - 128GB
4.	3 × RTX 2080 Ti - 12GB	12 × AMD Ryzen 2920X @ 3.50GHz - 128GB

Table B.1: Computing infrastructure.