Large Data-to-Text Generation

by

Varnan Sarangian

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Statistics

Waterloo, Ontario, Canada, 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis presents a domain-driven approach to sports game summarization, a specific instance of large data-to-text generation (DTG). We first address the data fidelity issue in the ROTOWIRE dataset by supplementing existing input records and demonstrating larger relative improvements compared to previously proposed purification schemes. As this method further increases the total number of input records, we alternatively formulate this problem as a multimodal problem (i.e. visual data-to-text), discussing potential advantages over purely textual approaches and studying its effectiveness for future expansion. We work exclusively with pre-trained end-to-end transformers throughout, allowing us to evaluate the efficacy of sparse attention and multimodal encoder-decoders in DTG and providing appropriate benchmarks for future work.

To automatically evaluate the statistical correctness of generated summaries, we also extend prior work on *automatic relation extraction* and build an updated pipeline that incorporates low amounts of human-annotated data which are quickly inflated via data augmentation. By formulating this in a "text-to-text" fashion, we are able to take advantage of LLMs and achieve significantly higher precision and recall than previous methods while tracking three times the number of unique relations. Our updated models are more consistent and reliable by incorporating human-verified data partitions into the training and evaluation process.

Acknowledgements

First and foremost, I would like to express profound gratitude to my supervisor, Professor Shoja Chenouri. His unwavering support, guidance, and patience throughout the past year have undoubtedly enriched my research and my skill-set as a novice in academia. I would also like to thank Dr. Liqun Diao and Dr. Aukosh Jagannath for taking the time to review this thesis. Finally, I would like to extend heartfelt thanks to my family and friends for their immense emotional support and encouragement. These past couple of years have presented unparalleled challenges to many across the globe. I am certainly fortunate to have had a solid support group who consistently kept me focused and disciplined.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data-to-text generation (DTG) (Reiter and Dale, 1997) is the task of translating a collection of non-linguistic data records (such as data tables) into coherent text *narratives*. One specific use case is *sports game summarization* where the narratives are orientated towards describing in-game events. It offers additional challenges on top of DTG due to the increased number of input records to select from and the complexity of the reference texts. However, prior work made use of datasets (Wiseman et al., 2017) that suffered from low data fidelity where a significant proportion of statements in the summaries are not accounted for in corresponding input records.

Wang (2019) proposed a purification scheme that removes ungrounded sentences in the summaries. This can be problematic in practice as the new summaries may no longer be coherent and the retained information becomes too trivial. In this thesis we propose an alternate domain-driven approach to *supplement* additional input records and preserve the original summaries. Doing so further extends the number of input records into unprecedented sizes, which we categorize as *large data-to-text generation*. We show that supplementing the input records yields a greater relative improvement than prior cleaning strategies, even without an exhaustive list of new relation types. We also curate an alternative approach of handling the large DTG problem by embedding the input data structures into images (i.e. *visual data-to-text generation*).

A related problem is to evaluate the statistical correctness of the summaries as standard language fluency metrics are insufficient. Wiseman et al. (2017) introduced a scheme to gauge statistical correctness by extracting and comparing relations between the predicted and gold summaries. The original extraction models were trained and benchmarked on noisy datasets generated in an algorithmic manner. We propose an updated pipeline that

incorporates human-annotated examples and data augmentation to build stronger and more reliable extraction models while tracking three times the number of unique relations. Furthermore, we formulate this in a "text-to-text" fashion and make use of generative language models for better generalization and easier post-processing.

The structure of the thesis is as follows: Chapter 2 provides a brief overview on various background topics such as statistical learning, neural networks, first-order optimization and regularization, the transformer architecture, language models, and transfer learning. Chapter 3 is a literature review on the evolution of transformers and data-to-text generation with respect to sports game summarization. Both directions are crucial to the contributions of this thesis which are discussed at length in Chapter 4. For readers familiar with the content of Chapter 2, we recommend starting from Chapter 3.

1.1 Contributions

To summarize, the contributions presented in this thesis are as follows:

- Propose a domain-driven approach to address the data fidelity issue in existing sports game summarization datasets. By supplementing existing input records, we note larger relative improvements across all metrics compared to the purification scheme of Wang (2019) while being more practical.
- Introduce a pipeline to automatically evaluate statistical correctness of the generated summaries in a "text-to-text" fashion using data augmentation. We achieve significantly higher results than those reported in (Wiseman et al., 2017; Puduppully et al., 2019*a*) despite incorporating three times the relations. Our models are also built off of human-verified data partitions ensuring that they are more grounded and reliable.
- Propose the sports summarization objective as a *visual data-to-text generation* problem, discuss several advantages over a purely textual approach, and study its effectiveness for potential expansion.
- Evaluate the efficacy of sparse attention and multimodal encoder-decoders on four novel large DTG datasets, providing appropriate benchmarks for future work.

Chapter 2

Background

This chapter provides a rudimentary tutorial on statistical learning, artificial neural networks, natural language processing, and their intersection. For additional information on related topics, please refer to (Goodfellow et al., 2016; Zhang et al., 2021; Hastie et al., 2009; James et al., 2013; Murphy, 2022; Prince, 2023).

2.1 Foundations of Statistical Learning

Statistical learning (or machine learning) is an interdisciplinary branch of statistics and computer science concerned with algorithms that extract patterns from data. There are further subdivisions broadly dictated by algorithmic outcome and data formulation, the most common of which being *supervised learning*.

2.1.1 Supervised Learning

The goal is to estimate an explicit mapping f between a *feature space* $\mathcal{X}$ and associated *target space* $\mathcal{Y}$ from a finite dataset of independent input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$. The inputs $\mathbf{x}_i$ are also known as *features*, *predictors* or *covariates* whereas the output $\mathbf{y}_i$ is the *label*, *target* or *response*. The dataset $\mathcal{D}$ used to construct the mapping is the *training set* with *sample size* n. In probabilistic terms, we wish to fit the conditional model $\mathbb{P}[\mathbf{y}|\mathbf{x}]$ over $\mathcal{D}$.

Supervised learning can be further divided based on the target space. For example, if $\mathcal{Y} = \{1, \dots, C\}$ is an unordered set of C disjoint *classes*, then f acts as a *decision rule*

that *classifies* the inputs into one (or more) categories. This is appropriately known as *classification*. On the other hand, *r*egression is the case where the outputs are real-valued quantities (i.e. $\mathcal{Y} = \mathbb{R}$).

In either case, the *hypothesis space* of f is restricted to some family of *parametric* models $\{f(\ \cdot\ ; \boldsymbol{\theta}) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ where $\boldsymbol{\theta}$ denotes the model-specific *parameters* and $\boldsymbol{\Theta}$ is the *parameter space*. With this restriction, the task of function estimation simplifies to finding the optimal parameter configuration $\widehat{\boldsymbol{\theta}}$.

2.1.2 Maximum Likelihood Estimation

The process of finding the optimal $\boldsymbol{\theta}$ from the training set is *model fitting* or *training*. This is formulated as

$$\widehat{\boldsymbol{\theta}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

where $\mathcal{L}(\boldsymbol{\theta})$ is some *loss* or *objective* function. For a finite sample, the most popular approach is to find the *point estimate* that is the most probable in light of the training data. This is the method of *maximum likelihood estimation* (MLE).

Assuming that the training observations are independently sampled from the same *data-generating distribution*, the MLE is

$$\widehat{\boldsymbol{\theta}}_{\text{mle}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{P}[\mathcal{D}|\boldsymbol{\theta}] = \prod_{i=1}^{N} \mathbb{P}[\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\theta}]$$

To ensure that this problem is compatible with modern optimization algorithms and finite floating point representations, we can equivalently minimize the conditional *negative log-likelihood* (NLL),

$$\widehat{\boldsymbol{\theta}}_{\text{mle}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \{-\log \mathbb{P}[\mathcal{D}|\boldsymbol{\theta}]\} = \operatorname*{argmin}_{\boldsymbol{\theta}} \left\{ -\sum_{i=1}^{N} \log \mathbb{P}[\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\theta}] \right\}$$

Under sufficient conditions (i.e. the data-generating distribution can be captured by the model family and $\boldsymbol{\theta}$ is identifiable), the MLE is consistent and *efficient* (Rao, 1945; Cramér, 1946). These statistical properties make MLE a favourable option for parameter estimation in general.

2.1.3 Empirical Risk Minimization

The NLL can be expressed as an expectation under the data-generating distribution, $\mathbb{E}_{\mathbb{P}_\mathcal{D}}[-\log \mathbb{P}_{\text{model}}(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})]$. If the loss is replaced with any general function $\ell(\mathbf{y}, f(\mathbf{x}; \boldsymbol{\theta}))$ where f is the model parameterized by $\boldsymbol{\theta}$, the empirical estimate of $\mathbb{E}_{\mathbb{P}_\mathcal{D}}[\ell(\mathbf{y}, f(\mathbf{x}; \boldsymbol{\theta}))]$ becomes

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \ell(\mathbf{y}_i, f(\mathbf{x}_i; \boldsymbol{\theta}))$$

This is called the *empirical risk*. In other words, maximum likelihood for fitting learning algorithms is a specific instance of *empirical risk minimization*.

2.1.4 Generalization

The true goal with learning algorithms however is to simultaneously minimize the expected loss on *unseen* data from the same distribution. Minimizing the objective over the training data doesn't guarantee such model *generalization*.

Suppose we denote the empirical risk over the training data, or *training error*, as $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_{\text{train}})$ and the *true* data-generating distribution $\mathbb{P}_\mathcal{D}$ is accessible to us. Instead of the empirical risk, one can compute the expectation of the loss directly (i.e. the *population risk*).

$$\mathcal{L}(\boldsymbol{\theta}; \mathbb{P}_\mathcal{D}) = \mathbb{E}_{\mathbb{P}_\mathcal{D}}[\ell(\mathbf{y}, f(\mathbf{x}; \boldsymbol{\theta}))]$$

The difference or *gap*, $\mathcal{L}(\boldsymbol{\theta}; \mathbb{P}_\mathcal{D}) - \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_{\text{train}})$, provides a measure of performance on unseen inputs. In reality, $\mathbb{P}_\mathcal{D}$ is unknown. However, by partitioning $\mathcal{D}$ into two subsets, the training and *test* sets, we can approximate the population risk empirically (i.e. *test error* or *generalization error*) using this new test set,

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_{\text{test}}) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{\text{test}}} \ell(\mathbf{y}, f(\mathbf{x}; \boldsymbol{\theta}))$$

One way to visualize the relationship between the generalization and training errors is to consider a plot with errors on the y-axis and model *capacity* on the x-axis. Informally, model capacity (or *complexity*) is its ability to fit various functions within a single family. Models with high capacity can fit a wide variety of different functions whereas models with low capacity are more limited.

In Figure 2.1, the training error goes to zero as model complexity increases, however the test error follows a *U-shaped curve*. For low complexity, the curves illustrates that

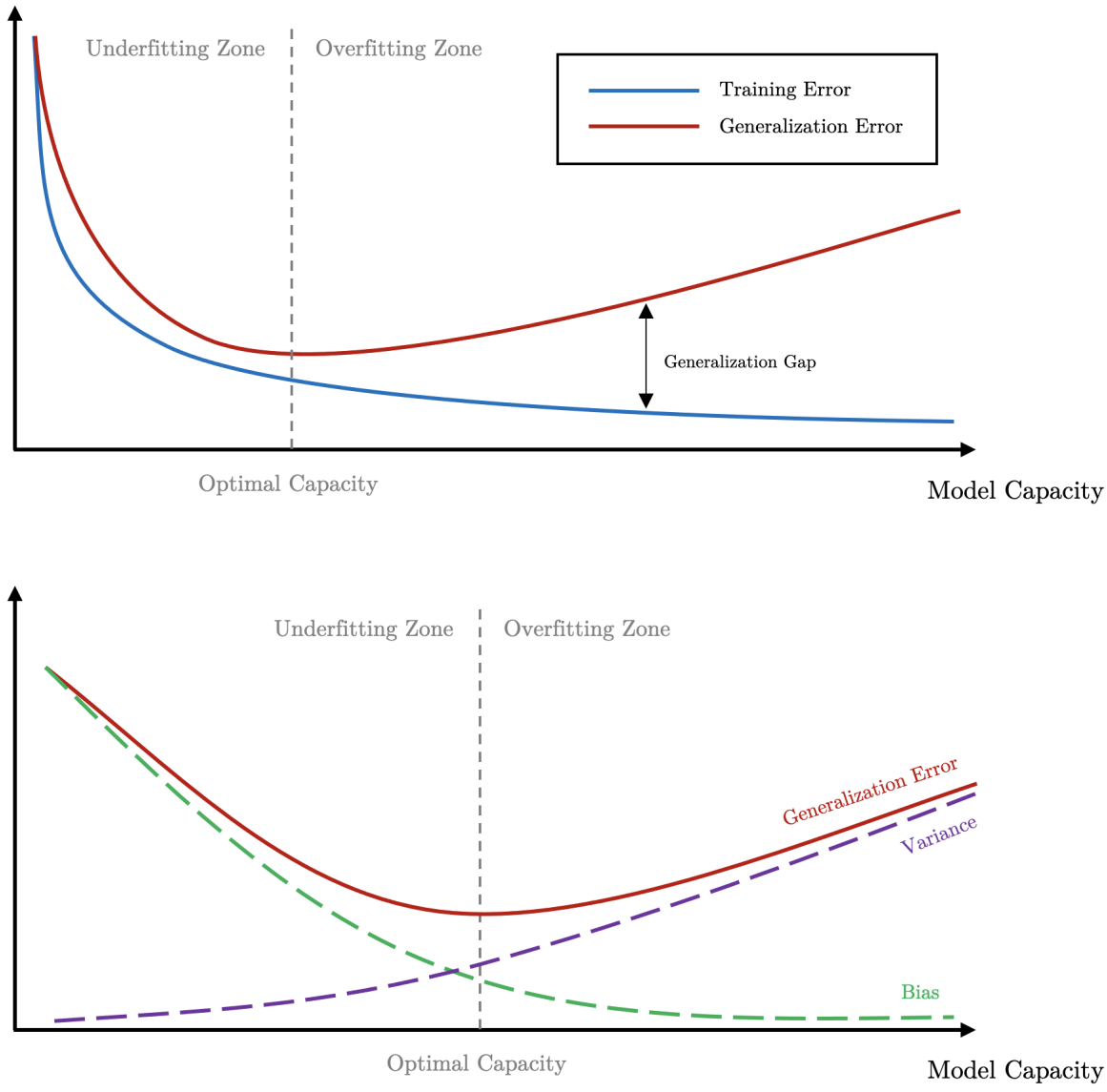


Figure 2.1: Two graphs illustrating the behaviour of various quantities and generalization as model capacity increases.

the model lacks the ability to sufficiently fit the training set. On the other side however, the model becomes too flexible such that it is prone to memorizing noise specific to the

training set. These phenomenons are known as *underfitting* and *overfitting* respectively[1]. Thus the *best* model is not the one that minimizes the empirical risk, but instead the test risk.

In practice, we need to have three partitions of the data set: training, test, and a *validation set*. The latter is used for model selection based on minimizing the "validation risk" (all of the previous intuition applies, however we use the validation risk in place of the test risk for model selection). The test or *hold-out* set is then used solely for estimating the population risk. It should not be used for model selection or refinement.

One final point is that no single model is optimal for multiple problems. This is coined as the *no free lunch theorem* (Wolpert, 1996). The reason for this is that a set of assumptions in one domain may not apply in another. The best model is then further contingent on domain knowledge on top of what was discussed earlier.

2.2 Feedforward Neural Networks

Artificial neural networks, or simply *neural networks*, are a family of highly-flexible models built using collections of connected simple computational units. The name comes the loose resemblance to its biological counterpart. These models are comprised of multiple layers that each apply transformations on their respective inputs to derive *latent representations*. *Deep learning* is the term used to describe neural networks with many layers.

The true power behind neural networks is their ability to automatically extract features. For example, to model a non-linear function using a linear model, one would need to first *manually transform* the input $\mathbf{x}$ into a pre-determined nonlinear representation ϕ. Deep learning instead *learns* ϕ through its intermediate layers which is then projected to the desired output with a final output layer. Simply put, $f(\mathbf{x}; \boldsymbol{\theta}, \mathbf{w}) = g(\phi(\mathbf{x}; \boldsymbol{\theta})^\top \mathbf{w})$ where ϕ is learned through $\boldsymbol{\theta}$, $\mathbf{w}$ is an additional *weight vector*, and g is the output layer.

2.2.1 Perceptron

The *perceptron* (Rosenblatt, 1958) was one of the earliest implementations of a neuron with "automated" learning. It was a linear binary classifier of the form

$$f(\mathbf{x}; \mathbf{w}) = \mathbb{1}\{\mathbf{w}^\top \mathbf{x} + b > 0\}$$

[1]This relationship is closely related to the *bias-variance trade off*. If the error is measured using the MSE, then increasing model capacity will typically decrease bias at the cost of increased variance.

where $\mathbf{w} \in \mathbb{R}^p$ is a *weight vector*, b is a *correction bias*, and the threshold indicator function $H(u) = \mathbb{1}\{u > 0\}$ is the *Heaviside step function*. This is also visualized in Figure 2.2.

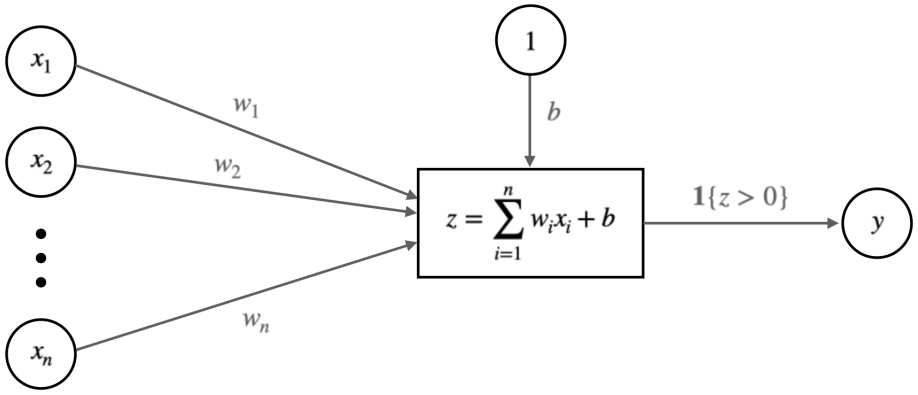


Figure 2.2: A visualization of the perceptron classifier. Importantly, note how the edges represent learnable weights and the nodes are values.

The *perceptron learning rule* was the explicit update-algorithm specifically for the perceptron to learn $\mathbf{w}$ from a random initialization. The idea is to update it only when the model makes an incorrect prediction. More accurately,

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t(\widehat{y}_j - y_j)\mathbf{x}_j \qquad b_{t+1} = b_t - \eta_t(\widehat{y}_j - y_j)$$

where $\mathbf{x}_j \in \mathbb{R}^p, y_j \in \{-1, 1\}$ is the sampled data point at iteration t and η_t is the step size.

However, Novikoff (1962) and Minsky and Papert (1969) demonstrated that the perceptron was only capable of modelling linear boundaries and converge for *linear separable* problems. Famously, the perceptron was unable to solve the *XOR problem* (Figure 2.3).

Despite this revelation, it was later shown that stacking multiple perceptrons into a connected *network* produced a new model with increased flexibility. That is, the capacity can be directly influenced by network *depth* and the number of units. This movement of *connectivity* (Goodfellow et al., 2016) laid the foundation for modern neural networks (Rumelhart and McClelland, 1986; McClelland et al., 1986).

2.2.2 Multilayer Perceptrons

Feedforward neural networks, or *multilayer perceptrons* (MLP), are stacked layers of perceptrons designed to propagate information in a unidirectional manner. Specifically, outputs

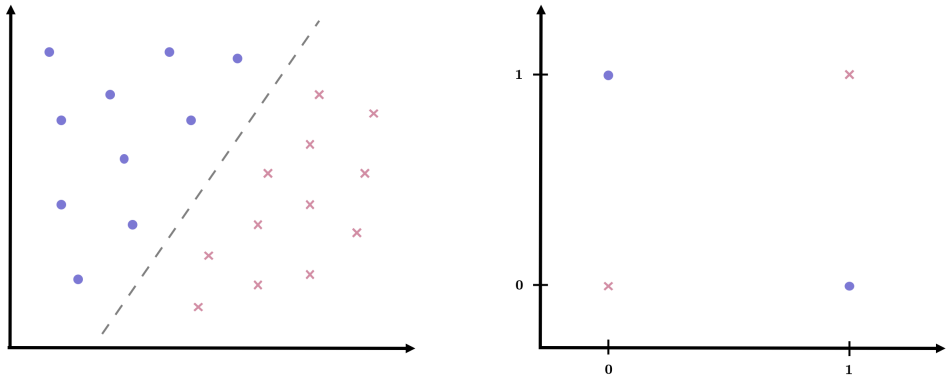


Figure 2.3: *Left:* An example of a *linear separable* problem where the decision boundary between the two classes is linear. *Right:* The XOR problem is a famous example of a non-linear separable problem.

of perceptrons from one layer act as inputs to perceptrons in the next layer. There are no recurrent connections where the model feeds intermediate outputs back into itself (i.e. directed acyclic graphs).

Deep neural networks (DNNs) are essentially a composition of many different functions. The model itself describes how these functions are composed together (see Figure 2.4). For example, the chain $f(\mathbf{x}) = f_3(f_2(f_1(\mathbf{x})))$ consists of the first *layer* f_1, second layer f_2, and so on. The overall length of the chain is the network *depth* and the final layer is the *output layer*.

The network is trained end-to-end where training examples specify the initial inputs and desired network output, but provide no information for guiding the intermediate layers. Since the outputs of these intermediate layers are unknown, they are referred to as *hidden layers*.

The original perceptron is modified to form the modern *artifical neuron/unit* used in all modern neural networks. The Heaviside function is replaced with a non-linear differentiable *activation function* $g : \mathbb{R} \to \mathbb{R}$ to allow for training via gradient-based methods (discussed in Section 2.4). The generalized unit becomes

$$y_i = g(z_i) = g(\mathbf{w}_i^\top \mathbf{x} + b)$$

where $z = \mathbf{w}^\top \mathbf{x} + b$ is the *pre-activation*, $\mathbf{w}, b$ are the parameters, and $\mathbf{x}$ are the input connections from preceding neurons.

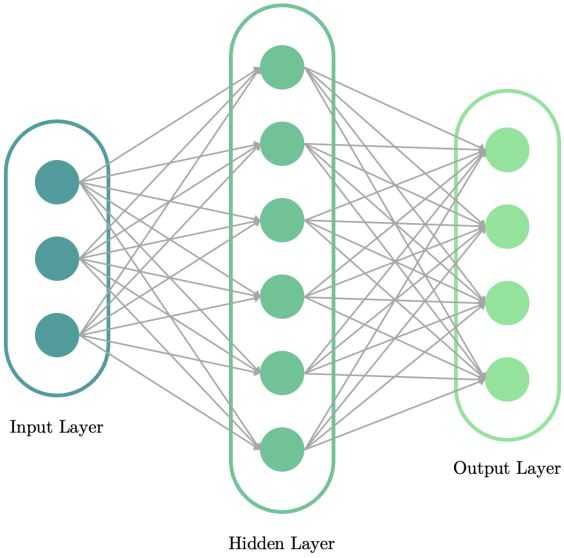


Figure 2.4: An example of a feedforward network with a depth of three (i.e. single hidden layer). Note that each layer is *fully-connected* so that each unit depends on every unit in the previous layer.

2.2.3 Output and Hidden Units

As discussed in 2.1.2, neural networks are trained using maximum likelihood where the objective is the NLL of the model distribution. Thus, the choice of the output unit's activation function is closely related to the cost function. If a feedforward network provides a hidden representation $\mathbf{h} = f(\mathbf{x}; \boldsymbol{\theta})$, the role of the output layer is to provide the final transformation over the pre-activation $\mathbf{W}^\top \mathbf{h} + \mathbf{b}$ to complete the network's task.

Some common examples include the identity activation to model the conditional multivariate Normal distribution $\mathbb{P}[\mathbf{y}|\mathbf{x}] = \mathcal{N}(\mathbf{y}; \widehat{\mathbf{y}}, \mathbb{I})$. For distributions such as Bernoulli and Multinomial (i.e. $\mathbb{P}[y = i|\mathbf{x}]$), the canonical activations are typically the sigmoidal and softmax functions respectively. These functions incorporate exponential terms which are approximately inverted by the cost function, which helps alleviate gradient *saturation* issues.

The choice of activation functions for intermediate or hidden units is unique to neural networks. Early adaptations of deep networks opted for sigmoid or hyperbolic tangent activations, though such units are prone to gradient saturation (Glorot and Bengio, 2010; Bengio, 2012) (especially without the NLL). The better choice is the *rectified linear unit*

(ReLU) (Fukushima, 1969) $g(z) = (z)_+ = \max\{0, z\}$, which was popularized by Jarrett et al. (2009), Nair and Hinton (2010), and Glorot et al. (2011). It is one of the simplest non-linearities where the gradient is one for $z > 0$. This contributes to stable and efficient training.

However, ReLU is prone to the *dying units* problem where units are no longer updated when their pre-activations are less than zero. Additional generalizations were later introduced to add a small non-zero slope for $z < 0$ (Maas, 2013; He et al., 2015*b*; Shang et al., 2016). Alternatively, smooth activations were proposed as solutions to the dying unit problem while simultaneously restricting gradient flow for large negative pre-activations (Glorot et al., 2011; Hendrycks and Gimpel, 2016; Clevert et al., 2015; Klambauer et al., 2017; Ramachandran et al., 2018; Howard et al., 2019). In practice, these variants only showed minor improvement over ReLU, if any.

2.2.4 Universal Approximation

In general, the *universal approximation theorem* (Hornik et al., 1989, 1990; Cybenko, 1989) states that an MLP with one hidden layer and intermediate "sigmoidal" activations can approximate any Borel measurable function to arbitrary accuracy (provided that the network has enough hidden units). Furthermore, the derivatives of the network approximate the derivatives of the function. Hornik (1991) later showed that this applies to a larger class of nonlinear activations.

This theorem guarantees the existence of an MLP that can represent the true function, but not if the solution is obtainable. For example, the optimization algorithm may not be able to find the correct parameter configuration or it could fit the wrong function through overfitting. The theorem also doesn't signal how large the network will be. In the worst case, the number of hidden units can be exponential (Barron, 1993) (i.e. one for each input configuration). Because of this, we typically resort to using deeper networks as they reduce the number of units while generalizing sufficiently. A *depth* version of the theorem was also proved by Lu et al. (2017).

2.2.5 Backpropagation

The celebrated *backpropagation* algorithm (Werbos, 1974; Bryson et al., 1969; Rumelhart et al., 1986*a*,*b*) computes the gradient of a loss function with respect to the parameters in

each network layer. We will only study its applications to MLPs here (specifically *reverse-mode differentiation*), however the general procedure can be applied to any DAG and is called *automatic differentiation (autodiff)* (Murphy, 2022).

Consider the MLP and associated loss function $\mathcal{L}(\mathbf{y}, \mathbf{o}) : \mathbb{R}^n \to \mathbb{R}$. Specifically,

$$\begin{aligned} \mathbf{o} = \mathbf{x}_L &= \mathbf{g}_L(\mathbf{W}_L \mathbf{x}_{L-1}) \\ \mathbf{x}_i &= \mathbf{g}_i(\mathbf{W}_i \mathbf{x}_{i-1}) \qquad i = 1, 2, \ldots, L-1 \end{aligned}$$

where $\mathbf{g}_i$ are the activation functions, $\mathbf{x}_0 = \mathbf{x}$ is the initial input, and the bias is incorporated into $\mathbf{W}$ by concatenating a one to each input $\mathbf{x}_i$. The main idea behind backpropagation is to use the chain rule to iteratively compute the gradients at each layer.

For example, with the output gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{o}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_L}$ the chain rule yields

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{W}_L} &= \frac{\partial \mathcal{L}}{\partial \mathbf{x}_L} \frac{\partial \mathbf{x}_L}{\partial \mathbf{W}_L} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{L-1}} &= \frac{\partial \mathcal{L}}{\partial \mathbf{x}_L} \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_{L-1}} \frac{\partial \mathbf{x}_{L-1}}{\partial \mathbf{W}_{L-1}} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{L-2}} &= \frac{\partial \mathcal{L}}{\partial \mathbf{x}_L} \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_{L-1}} \frac{\partial \mathbf{x}_{L-1}}{\partial \mathbf{x}_{L-2}} \frac{\partial \mathbf{x}_{L-2}}{\partial \mathbf{W}_{L-2}} \\ \vdots \quad &= \quad \vdots \end{aligned}$$

which can be efficiently calculated by traversing backwards through the network after a *forward propagation* and reusing gradients from higher layers.

2.3 Regularization

Regularization is *any modification to a learning algorithm that is intended to reduce its generalization error but not its training error* (Goodfellow et al., 2016). Simply put, the idea is to increase the bias of the estimator by restricting the model capacity in hopes of reducing the variance and preventing overfitting. This is crucial for modern neural networks that are significantly over-parameterized (i.e. millions or billions of parameters).

2.3.1 Parameter Norm Penalties

A common approach to limiting model capacity is incorporating an additional parameter norm penalty $\Omega(\boldsymbol{\theta})$ to the objective. Note that $\boldsymbol{\theta}$ denotes the set of all applicable parameters

in the models. The new objective to optimize becomes

$$\tilde{\mathcal{L}}(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) = \mathcal{L}(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) + \alpha\Omega(\boldsymbol{\theta})$$

where $\alpha \geq 0$ is a hyperparameter controlling the amount of regularization. In practice, the penalty is typically applied only to the *weights* of the affine transformation at each layer and excludes the biases. It is also possible to use a different penalty for each layer of the network, however a single penalty is normally sufficient.

The simplest penalty is the L_2 norm $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{\theta}||_2^2$, also known as *weight decay* (this same penalty is adapted for linear models in ridge regression). Parameters that contribute significantly to reducing the original objective are relatively preserved whereas the rest are decayed to zero.

Another option is the L_1 norm $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_1 = \sum_i |\theta_i|$. This penalty results in a *sparse* solution that forces weights to be exactly zero. In contrast, the L_2 norm keeps nonzero weights as nonzero. The sparsity property acts as a *feature selection* mechanism, (i.e. LASSO regression (Tibshirani, 1996)), where weights set to zero indicate that the corresponding features can be discarded.

2.3.2 Dropout

Dropout (Srivastava et al., 2014) is a regularization strategy designed specifically for neural networks. The idea is to randomly turn off all the outgoing connections from each neuron with probability p (on a per-example basis), as shown in Figure 2.5. This dramatically reduces overfitting as it prevents the network from learning complex and fragile relationships between units.

More formally, we are estimating a noisy version of the weights $\theta_{l,i,j} = w_{l,i,j}\epsilon_{l,i}$ where $\epsilon_{l,i} \sim \text{Bernoulli}(1-p)$. During inference, the noise is usually disabled. To ensure that the weights have the same expectation at test time as they did during training, we set $w_{l,i,j} = \theta_{l,i,j}\mathbb{E}[\epsilon_{l,i}]$ where $\mathbb{E}[\epsilon_{l,i}] = 1 - p$. Another interpretation is that dropout produces an approximation of *ensembling* many neural networks by training an ensemble of different sparse subnetworks.

2.3.3 Data Augmentation

The best approach to improving a model's ability to generalize is to train it over a large diverse training set. As this is seldom the case, one can alternatively create artificial

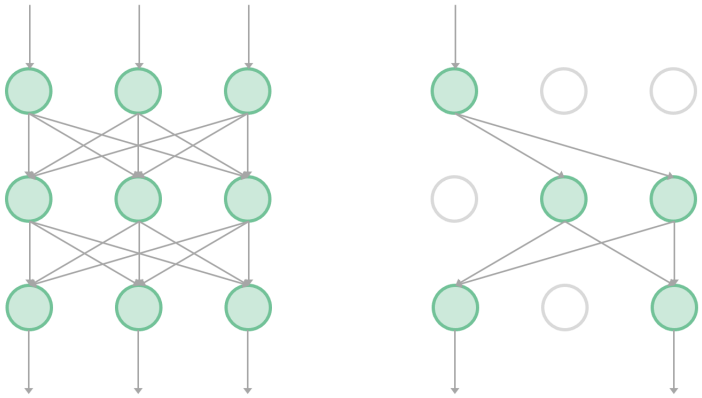


Figure 2.5: An example of a feedforward network (*left*) with dropout applied (*right*). The units are disabled probabilistically (per-example) such that only a subnetwork is propagated.

examples to arbitrarily inflate the amount of training data. This is easy to see for a task such as *image classification* where the model needs to be invariant to a wide spectrum of transformations (i.e. rotation, translations, crops, blurs, etc.). An "infinitely" large training set can then be generated by randomly transforming the input $\mathbf{x}$ from the pair $(\mathbf{x}, y)$ for every batch.

For text inputs, task-agnostic augmentation is more closely aligned to noise injection. For example, some augmentation strategies at the discrete level include replacing tokens, removing characters, permute word or sentence order, etc. Similarly, one can add noise by incorporating a small Gaussian error into the corresponding word embeddings (Section 2.6.2). Refer to (Shorten et al., 2021; Feng et al., 2021; Chen et al., 2021; Wei and Zou, 2019) for a comprehensive list of general augmentation schemes for text inputs.

2.3.4 Other Regularizers

For classification tasks, *label smoothing* redistributes the probability mass of the target labels. For example, we replace the *hard* 0 and 1 targets in a softmax (with k values) with $\frac{\epsilon}{k-1}$ and $1-\epsilon$ respectively. This prevents the softmax layer from learning large weights (since it can never converge to 0 or 1).

Early stopping is the basic heuristic of terminating model training when the validation error starts to increase. This can be thought of restricting the model capacity by reducing

the number of training iterations, effectively cutting off the model before it can overfit.

2.4 Gradient-based Optimization

Neural networks are typically trained with first-order gradient-based optimization algorithms. Due to non-convex objectives and overparameterized networks, the loss landscape will contain many local minima. As a result, we often settle for finding f^* such that the loss is sufficiently low, but likely not globally minimal.

To motivate the next few algorithms, consider a function with multiple inputs $f : \mathbb{R}^n \to \mathbb{R}$. The *directional derivative* is the slope of f in the direction provided by some specified unit vector $\mathbf{u}$. That is,

$$\frac{\partial}{\partial \alpha} f(\mathbf{x} + \alpha \mathbf{u}) \bigg|_{\alpha=0} = \mathbf{u}^\top \nabla_{\mathbf{x}} f(\mathbf{x})$$

To minimize f, we need to find the direction that f decreases the fastest. Notice that

$$\min_{\mathbf{u}, ||\mathbf{u}||_2=1} \mathbf{u}^\top \nabla_{\mathbf{x}} f(\mathbf{x}) = \min_{\mathbf{u}, ||\mathbf{u}||_2=1} ||\mathbf{u}||_2 ||\nabla_{\mathbf{x}} f(\mathbf{x})||_2 \cos\theta$$

where θ is the angle between $\mathbf{u}$ and the gradient. Since $\mathbf{u}$ has unit norm and the gradient is free of $\mathbf{u}$, we are directly optimizing $\cos\theta$. This is clearly minimized when $\mathbf{u}$ is in the opposite direction of the gradient.

This is the *method of steepest descent* or simply *gradient descent*. The algorithm proposes updating a point iteratively with

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \eta_i \nabla_{\mathbf{x}} f(\mathbf{x})$$

where the *learning rate* $\eta_i > 0$ determines the step size at iteration i.

2.4.1 Stochastic and Mini-batch Gradient Descent

Large datasets are ideal for training neural networks, however they are difficult to work with directly. To see this, note that the objective often decomposes into a sum of per-example losses.

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} \ell(\mathbf{y}, f(\mathbf{x}; \boldsymbol{\theta}))$$

This implies that $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ also decomposes into a sum as differentiation is linear. The computational cost is $O(n)$, which can be intractable for larger training sets.

Stochastic gradient descent (SGD) is an extension of the gradient descent algorithm (following from a class of *stochastic optimization* problems) that treats this sum as an expectation $\nabla_{\boldsymbol{\theta}}\mathcal{L} = \mathbb{E}[\nabla_{\boldsymbol{\theta}}\ell]$. Doing so allows one to empirically estimate the gradient using a small subset of examples. Specifically, at each step of the algorithm we sample a *minibatch* $\{\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(m')}\}$ drawn uniformly from the training set. The size m' is generally kept small and fixed regardless of the size of the training set. The gradient estimate becomes

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) \approx \frac{1}{m'}\sum_{i=1}^{m'}\nabla_{\boldsymbol{\theta}}\ell(\mathbf{x}_i, y_i, \boldsymbol{\theta})$$

which is used to then update $\boldsymbol{\theta}_i$ instead of the full gradient.

SGD formally refers to the extreme case where $m' = 1$, and the algorithm described here is *minibatch gradient descent*. However, it is common to refer to any of these algorithms as simply *stochastic*. Minibatches are selected randomly to ensure that the estimate is unbiased (i.e. i.i.d. examples). In practice, the full dataset is shuffled once at the start of training. It is also common practice to make several passes over the dataset (called *epochs*) as the benefit of decreasing the training error can offset the harm of increasing the generalization gap.

Gradient accumulation is a technique to train larger DNNs by aggregating the gradients from multiple batches before updating the parameters. This simulates the effect of a larger *effective batch size* without the computational burden by incorporating information from more examples into the gradient. For example, with a batch size of 8 and gradient accumulation of 4, the model processes four batches and accumulates the gradients before updating its parameters (analogous to using an effective batch size of 32).

2.4.2 Momentum

Momentum (Bertsekas, 1995) is a method designed to accelerate learning in SGD along problematic regions in the loss curvature. The idea is to accumulate an *exponentially weighted moving average* (EWMA) of prior gradients to encourage the algorithm to continue moving along previously good directions. The update steps are

$$\mathbf{m}_t = \beta\mathbf{m}_{t-1} + (1-\beta)\mathbf{g}_{t-1} \qquad \boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t\mathbf{m}_t$$

where $\mathbf{m}_t$ is the momentum, $\mathbf{g}_{t-1}$ is shorthand for the gradient, and $0 < \beta < 1$ ($\beta = 0$ recovers SGD). Momentum can potentially simulate the effects of a larger minibatch as it incorporates historical information from earlier batches into the next direction.

An issue with the current momentum update is that it is unable to slow down in some regions. *Nesterov momentum* (Nesterov, 2004) modifies the update to include an extrapolation step.

$$\mathbf{m}_{t+1} = \beta\mathbf{m}_t - \eta_t \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t + \beta\mathbf{m}_t) \qquad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \mathbf{m}_{t+1}$$

This acts as a *look ahead* which is later corrected. The intuition is that the momentum vector is already pointing roughly towards the optimal direction, so the gradient at the new location can be more accurate than the current location.

2.4.3 Choosing the Learning Rate

As SGD introduces estimation noise, it is crucial to try and counteract this with a decreasing learning rate. For full batch gradient descent (i.e. batch is the full dataset), the gradient behaves as intended so a constant learning rate is sufficient. The sequence of resulting learning rates at each time step $\{\eta_i\}$ is called the *learning rate schedule*. The choice of η_0 is also very important. A large value will result in violent oscillations whereas a low value will slow down or freeze learning prematurely.

Some common schedules are *piecewise constant* $\eta_t = \eta_i$ if $t_i \leq t \leq t_{i+1}$, *exponential decay* $\eta_t = \eta_0 e^{-\lambda t}$, and *polynomial decay* $\eta_t = \eta_0(\beta t + 1)^{-\alpha}$. Thresholds can also be computed adaptively. For example, we can decrease the step size by some factor λ when the validation loss plateaus (this is called *reduce-on-plateau*).

In deep learning, it is also common to introduce a *learning rate warmup* (Goyal et al., 2017) or *one-cycle learning rate schedule* (Smith, 2018). The idea is to quickly increase the learning rate from zero for some number of steps before applying a decreasing schedule. This is to stabilize initial parameter updates if the loss landscape is poorly conditioned. A slower learning rate allows for the algorithm to discover flatter regions which it can then traverse with larger steps.

2.4.4 Adaptive Learning Rates

Rather than using a single learning rate for all parameters, it may be more desirable to use a separate rate for each parameter and automatically adapt the rates throughout

training. More formally, these are examples of *preconditioned SGD* (Murphy, 2022) where the parameter update step becomes

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \eta_i \mathbf{M}_i^{-1} \mathbf{g}_i$$

and $\mathbf{M}_i$ is the *preconditioner*. Note that if the preconditioner is the Hessian then we recover *Newton's Method* which utilizes second-order information. However, the Hessian is difficult to estimate and expensive for larger models such as DNNs. Instead, $\mathbf{M}_i$ is designed to further speedup SGD (possibly on top of momentum).

AdaGrad or "adaptive gradient" (Duchi et al., 2010) re-scales all of the parameters by the inverse root sum of their previous gradients squared. That is,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \frac{1}{\sqrt{\mathbf{s}_t + \epsilon}} \odot \mathbf{g}_t \qquad \mathbf{s}_t = \mathbf{s}_{t-1} + \mathbf{g}_t \odot \mathbf{g}_t$$

where division and square root is element-wise and $\epsilon > 0$ is a small term to avoid dividing by zero. This encourages sparsely updated parameters to have larger future step sizes while decaying frequently updated parameters. The benefit here is that AdaGrad eliminates the need to manually adjust the learning rate, thus acting as an *adaptive learning rate*.

However, the accumulation of squared gradients from the start of training can prematurely diminish the effective learning rate. An alternative is to substitute the squared sum with the EWMA,

$$s_{t+1,d} = \beta s_{t,d} + (1-\beta) g_{t,d}^2$$

Since $\sqrt{s_{t,d}} \approx \text{RMS}(\mathbf{g}_{1:t,d}) = \sqrt{\sum_{\tau=1}^{t} g_{\tau,d}^2 / t}$ where RMS means "root mean squared," the method is called *RMSProp* (Tieleman and Hinton, 2012).

The *AdaDelta* method (Zeiler, 2012) was introduced independently and is similar to RMSProp, but they also keep an EWMA of past update steps $\boldsymbol{\delta}_t$ to keep the learning rate *dimensionless*. That is,

$$\begin{aligned} \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \Delta\boldsymbol{\theta}_t = \boldsymbol{\theta}_t - \eta_t \frac{\sqrt{\boldsymbol{\delta}_t + \epsilon}}{\sqrt{\mathbf{s}_t + \epsilon}} \odot \mathbf{g}_t \\ \mathbf{s}_t = \beta \mathbf{s}_{t-1} + (1-\beta) \mathbf{g}_t \odot \mathbf{g}_t, \qquad \boldsymbol{\delta}_t = \beta \boldsymbol{\delta}_{t-1} + (1-\beta)(\Delta\boldsymbol{\theta}_t)^2 \end{aligned}$$

It is also possible to combine RMSProp with momentum to yield the parameter update,

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t, \quad \mathbf{s}_t = \beta_2 \mathbf{s}_{t-1} + (1-\beta_2)\mathbf{g}_t^2, \quad \Delta\boldsymbol{\theta}_t = -\eta_t \frac{1}{\sqrt{\mathbf{s}_t + \epsilon}} \mathbf{m}_t$$

This was proposed as *Adam* (Kingma and Ba, 2015), which stands for *adaptive moment estimation*. Notice that if $\mathbf{m}_0 = \mathbf{s}_0 = \mathbf{0}$, then the initial estimates will be biased towards smaller values (a common issue with EWMA in general). The authors proposed using bias-corrected moments to circumvent this

$$\widehat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-\beta_1^t}, \quad \widehat{\mathbf{s}}_t = \frac{\mathbf{s}_t}{1-\beta_1^t}, \quad \Delta\boldsymbol{\theta}_t = -\eta_t \frac{1}{\sqrt{\widehat{\mathbf{s}}_t + \epsilon}} \widehat{\mathbf{m}}_t$$

Adafactor (Shazeer and Stern, 2018) is a memory-efficient extension of Adam using relative step sizes, clipped RMS, remove the first moment estimator and rewrite the second moment as an adaptive function of the step t. Furthermore, they only maintain the per-dimension sums of the squared gradients and approximate the full sum to reduce memory consumption during optimization.

2.4.5 Parameter Initialization

The weight matrices in a DNN are initialized randomly to ensure that there is no symmetry between intermediate units. The specific distribution is important for stable learning however. For example, Glorot and Bengio (2010) showed that sampling from $\mathcal{N}(0, \sigma^2)$ for fixed σ can result in exploding activations. To prevent this, they propose restricting the output variance to $\sigma^2 = 2/(n_{\text{in}} + n_{\text{out}})$ where n_{in} is the *fan-in* (number of incoming connections) and n_{out} is the *fan-out* (number of outgoing connections). This is known as the *Xavier* or *Glorot initialization*. For the special case where $\sigma^2 = 2/n_{\text{in}}$, this becomes the *He initialization* (He et al., 2015*b*). It isn't necessary to use a Normal distribution, for example one can also sample from Uniform$(-a, a)$ where $a = \sqrt{6/(n_{\text{in}} + n_{\text{out}})}$.

2.4.6 Network Layers for Optimized Learning

Under gradient optimization, parameters are updated under the assumption that all other layers are fixed. In practice they are updated simultaneously which can produce unexpected results or destabilise training altogether (i.e. vanishing/exploding gradients).

One solution is to incorporate additional *normalization layers* throughout the network to standardize the statistics of the hidden units before passing them through the next layer. *Batch normalization* (Ioffe and Szegedy, 2015) standardizes the activation distribution to zero mean and unit variance, when averaged across the minibatch. Specifically for an activation vector $\mathbf{z}_n$,

$$\tilde{\mathbf{z}}_n = \boldsymbol{\gamma} \odot \widehat{\mathbf{z}}_n + \boldsymbol{\beta}, \quad \widehat{\mathbf{z}}_n = \frac{\mathbf{z}_n - \mu_{\mathcal{B}}}{\sqrt{\boldsymbol{\sigma}_{\mathcal{B}}^2 + \epsilon}}, \quad \boldsymbol{\mu}_{\mathcal{B}} = \frac{1}{|\mathcal{B}|}\sum_{\mathbf{z}\in\mathcal{B}} \mathbf{z}, \quad \boldsymbol{\sigma}_{\mathcal{B}}^2 = \frac{1}{|\mathcal{B}|}\sum_{\mathbf{z}\in\mathcal{B}} (\mathbf{z} - \boldsymbol{\mu}_{\mathcal{B}})^2$$

where $\boldsymbol{\mu}_{\mathcal{B}}, \boldsymbol{\sigma}^2_{\mathcal{B}}$ are the batch statistics and $\boldsymbol{\beta}, \boldsymbol{\gamma}$ are additional parameters. During inference, $\boldsymbol{\mu}_l, \boldsymbol{\sigma}^2_l$ for layer l are computed across the full training set and frozen for use.

Batch normalization can significantly stabilize training and allow for larger learning rates, however it is sensitive to small batch sizes. An alternative approach is *layer normalization* (Ba et al., 2016) which pools the statistics over the tensor dimensions. That is, for a hidden layer $\mathbf{h}$ with dimensionality d_h,

$$\tilde{\mathbf{h}} = \gamma\widehat{\mathbf{h}} + \beta, \qquad \widehat{\mathbf{h}} = \frac{\mathbf{h} - \mu}{\sqrt{\sigma^2 + \epsilon}}, \qquad \mu = \frac{1}{d_h}\sum_{i=1}^{d_h} h_i, \qquad \sigma^2 = \frac{1}{d_h}\sum_{i=1}^{d_h}(h_i - \mu)^2$$

Outside of normalization, another design strategy is to add *residual/skip connections* (He et al., 2015*a*) that bypass intermediate layers to send information from a lower layer directly to a higher layer (see Figure 2.6). This reduces the length of the shortest path and improves information flow, allowing for easier gradient propagation. It was also shown that the loss surfaces of networks with residual connections tend to be smoother around minima (Li et al., 2017).

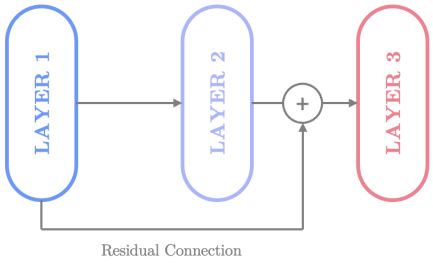


Figure 2.6: An example of a residual connection that sends the output of an earlier to a later layer while skipping an intermediate layer. This is normally just added to the final output of the layer(s) that are skipped.

2.5 Transformers

Transformers (Vaswani et al., 2017) are special neural architectures designed to process sequences. They are reminiscent of deep feedforward networks and are easy to scale and parallelize. These architectures are the forefront of modern deep learning success. This section focuses solely on the original architectural components with applications and advancements discussed in Section 3.1.

2.5.1 Self-Attention

Self-attention (Cheng et al., 2016; Graves, 2013; Bahdanau et al., 2014), also called *intra-attention*, is a mechanism that dynamically relates different positions of a sequence to determine their contextual significance to one another.

More formally, consider the input sequence $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_n$ and real-valued score function $a : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$. Self-attention maps $\mathbf{X}$ to a new sequence of the same size $\mathbf{Y} = \mathbf{y}_1, \ldots, \mathbf{y}_n$ where

$$\mathbf{y}_i = \sum_{j<i} \alpha_{ij} \mathbf{x}_j, \quad \alpha_{ij} = \frac{\exp\left(a(\mathbf{x}_i, \mathbf{x}_j)\right)}{\sum_{k<i} \exp\left(a(\mathbf{x}_i, \mathbf{x}_k)\right)}, \; j = 1, \ldots, i-1$$

Essentially, each $\mathbf{y}_i$ is a weighted combination of its inputs where the weights are calculated based on some scoring function relating $\mathbf{x}_i$ to every available input. The softmax function normalizes the n scores into a valid distribution. This operation is visualized in Figure 2.7. For applications such as *autoregressive generation*, the mechanism is restricted to preceding

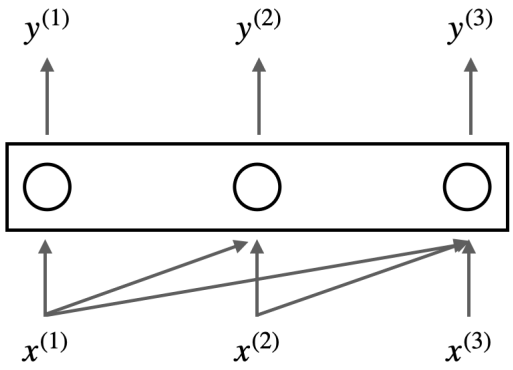


Figure 2.7: Visualization of the basic self-attention mechanism where $\mathbf{y}_i$ is a function of all the inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_i$.

inputs during training (i.e. *causal self-attention*). Similarly for applications where the full input sequence is needed then self-attention is unrestricted. These use cases are further discussed in Section 2.5.4.

Self-Attention as a Parameterized Dictionary Lookup

Transformers extend on the basic self-attention mechanism by providing an interpretation of the different vectors as a dictionary lookup system. Specifically,

- The *query* $\mathbf{q}$ is the *current focus* of the attention when being compared to all of the other preceding inputs.
- The *key* $\mathbf{k}$ is a preceding input that is compared to the query.
- The *value* $\mathbf{v}$ is the vector used to compute the output from the key.

To capture these roles, each $\mathbf{x}_i$ is projected using different weight matrices based on the role. For example,

$$\mathbf{q}_i = \mathbf{W}^Q \mathbf{x}_i \qquad \mathbf{k}_i = \mathbf{W}^K \mathbf{x}_i \qquad \mathbf{v}_i = \mathbf{W}^V \mathbf{x}_i$$

where the dimensions of all three roles match (i.e. $d_q = d_k = d_v$). The updated self-attention operation using the projected vectors is

$$\mathbf{y}_i = \sum_{j<i} \alpha_{ij} \mathbf{v}_j, \quad \alpha_{ij} = \frac{\exp\left(a(\mathbf{q}_i, \mathbf{k}_j)\right)}{\sum_{r<i} \exp\left(a(\mathbf{q}_i, \mathbf{k}_r)\right)}, \; j = 1, \ldots, i-1$$

Scoring functions can vary in sophistication from vector similarity to parameterized functions (Luong, Le, Sutskever, Vinyals and Kaiser, 2015; Graves et al., 2014; Bahdanau et al., 2014). Transformers use a *scaled dot-product* $(\mathbf{q}_i \cdot \mathbf{k}_j)/\sqrt{d_k}$ (Vaswani et al., 2017) to nurture large magnitude scores before exponentiating. This updated version of self-attention is visualized in Figure 2.8.

Note on Efficient Implementation

This operation can be further compacted as matrix products for efficiency. Suppose $\mathbf{X} \in \mathbb{R}^{N \times d}$ represents the sequence of N input vectors. The full self-attention operation becomes

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \; \mathbf{K} = \mathbf{X}\mathbf{W}^K, \; \mathbf{V} = \mathbf{X}\mathbf{W}^V, \quad \text{SelfAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

Since $\mathbf{Q}\mathbf{K}^\top$ calculates all pairwise scores, the upper-triangular elements (i.e. non-preceding inputs) are set to $-\infty$ before applying the softmax. Note that self-attention has quadratic compute and memory complexity $O(N^2)$ against sequence length. Efficient variants that aim to reduce these constraints is an ongoing research direction (see Section 3.1.1).

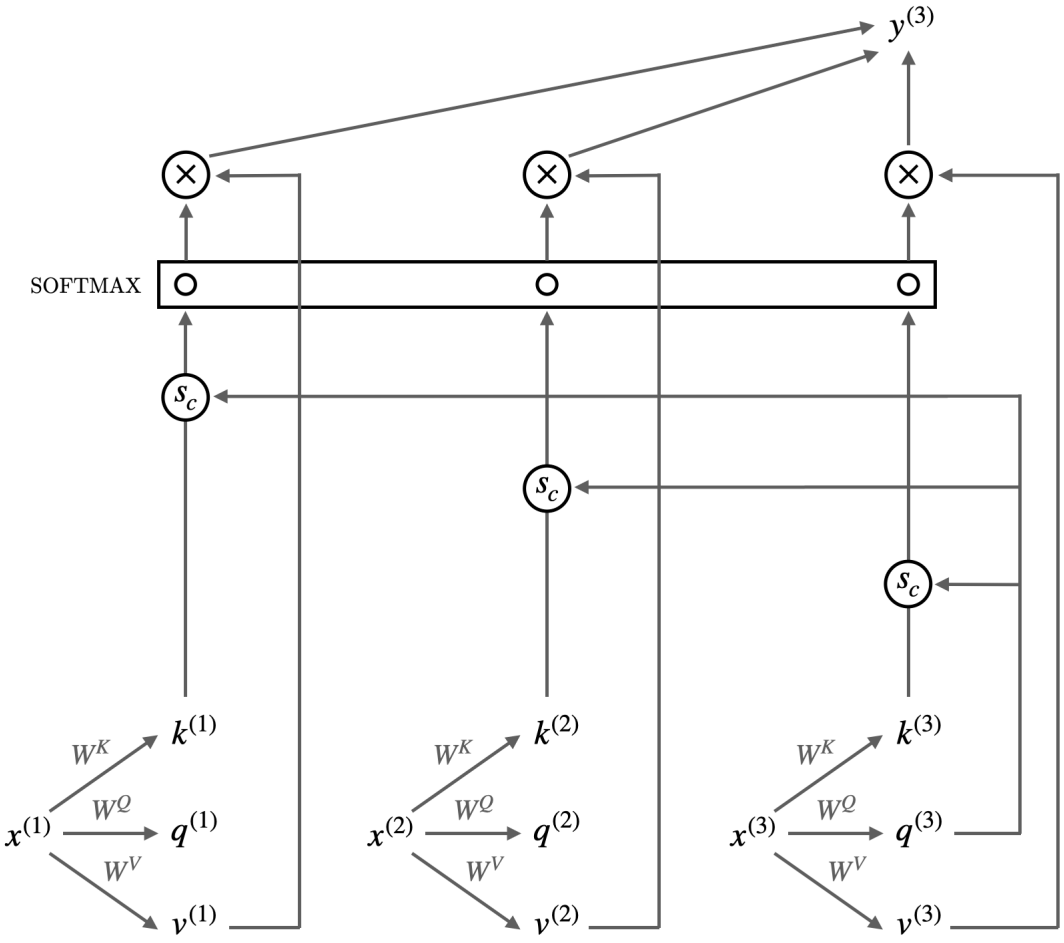


Figure 2.8: Visualization of the self-attention mechanism in Transformers for a single head (see Section 2.5.2). Note the relationships and roles each of the project key, query, and value vectors take. The function s_c denotes the scoring function between the query and key.

2.5.2 Multi-Headed Attention

For rich inputs such as text, there can be many different ways individual words relate to one another simultaneously. It is difficult to capture all these distinct parallel relations within a single self-attention layer. To address this, Transformers use *multi-headed self-attention* which is are sets of independent self-attention layers (each called a *head*) that

reside in parallel. Each head learns different patterns that exist among the inputs within same level of abstraction.

Each head i has its own key, query, and value matrices $\mathbf{W}_i^K, \mathbf{W}_i^Q, \mathbf{W}_i^V$. The projections are done separately for each head, but the remaining operations are the same. The key and query embeddings should share the same size d_k, however the value d_v may differ. The outputs of all h heads are concatenated and reduced down to the original input dimension d using $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d}$. The full calculation is summarized as

$$\text{MultiHeadAttn}(\mathbf{X}) = (\text{head}_1 \oplus \cdots \oplus \text{head}_h)\mathbf{W}^O \qquad \mathbf{Q}_i = \mathbf{X}\mathbf{W}_i^Q, \mathbf{K}_i = \mathbf{X}\mathbf{W}_i^K, \mathbf{V}_i = \mathbf{X}\mathbf{W}_i^V$$
$$\text{head}_i = \text{SelfAttention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i)$$

This layer can be substituted in any place where there is a single self-attention layer.

2.5.3 Transformer Blocks

A *transformer block* is a combination of multiheaded attention, feedforward layers, residual connections, and normalization layers where the input and output dimensions are matched to allow for easy scaling (visualized in Figure 2.9). A transformer itself is just stack of individual transformer blocks. The feedforward layer is applied to each input in the sequence through *shared weights*. For regularization, dropout (Srivastava et al., 2014) is applied within the feedforward layer, on the skip connection, on the attention weights, and at the input and output.

Positional Encoding

There is no notion of relative or absolute positions for inputs to any attention mechanism. One solution is to modify the input by combining dedicated *positional embeddings* that are unique to each position. Typically a fixed number of positions are defined and learned along with the other parameters. These embeddings are then added to $\mathbf{X}$ (or possibly before a subset of the key/query/value projections).

An alternative approach is to use a *static* function that maps integer inputs to real-value vectors such that they capture inherit relationships among the positions (i.e. position 4 is closer to 5 than 17). Vaswani et al. (2017) proposed such a mechanism by combining sinusodal functions at different frequencies.

Rather than absolute positions, relative position embeddings (Shaw et al., 2018; Huang et al., 2019) directly model offsets between query position i and key position j. These

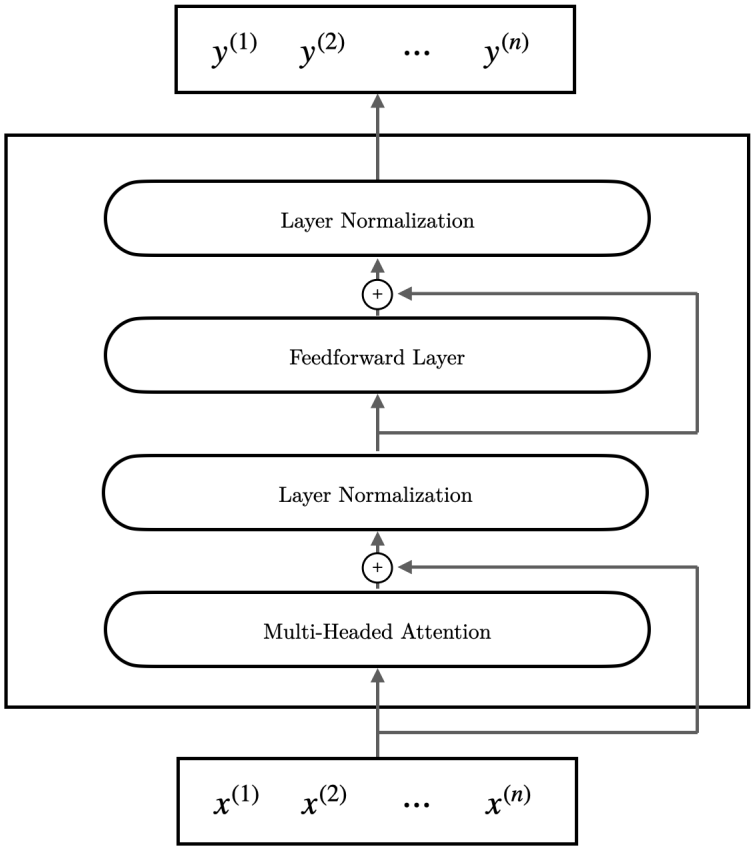


Figure 2.9: The canonical Transformer block as presented in Vaswani et al. (2017). A transformer is just multiple blocks stacked together in succession to develop deep representations of the input sequence.

embeddings are learned for every offset (i.e. $\beta_{i,j}$) and used to directly modify the attention matrix.

2.5.4 Encoder-Decoders

The transformer was originally designed for *neural sequence transduction* tasks (i.e. mapping input sequences to output sequences of variable length) such as machine translation (Bahdanau et al., 2014; Cho et al., 2014; Sutskever et al., 2014). The canonical

architecture for such tasks is an *encoder-decoder* where the encoder transforms the input $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ to a latent representation $\mathbf{H}^{\text{enc}} = (\mathbf{h}_1, \ldots, \mathbf{h}_n)$ which the decoder uses to autoregressively generate the output $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_m)$.

For the transformer encoder-decoder, the encoder is a stack of $N = 6$ transformer blocks as described in Section 2.5.3. As the encoder is normally allowed to incorporate the full sequence based on the task, the upper-triangular entries of the attention matrix are not masked.

Similarly, the decoder is also a stack of $N = 6$ blocks but with masked attention for conditioning on its previous outputs. Furthermore, the decoders' blocks include a dedicated *cross-attention* (or *encoder-decoder/source attention*) layer that attends to both the encoders' and previous layer's representations. More formally, $\mathbf{H}^{\text{enc}}$ is projected using dedicated weights $\mathbf{W}^K_{\text{enc}}, \mathbf{W}^V_{\text{enc}}$ and the prior decoder layer's representation $\mathbf{H}^{\text{dec}}_{i-1}$ is projected with $\mathbf{W}^Q_i$. The cross attention head is computed as

$$\mathbf{Q}_i = \mathbf{H}^{dec}_{i-1}\ \mathbf{W}^Q_i, \quad \mathbf{K}_{enc} = \mathbf{H}^{enc}\ \mathbf{W}^K_{enc}, \quad \mathbf{V}_{enc} = \mathbf{H}^{enc}\ \mathbf{W}^V_{enc}$$

$$\text{CrossAttention}(\mathbf{Q}_i, \mathbf{K}_{enc}, \mathbf{V}_{enc}) = \text{softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}^\top_{enc}}{\sqrt{d_k}}\right)\mathbf{V}_{enc}$$

The full cross-attention layer is implemented with multiple heads (Section 2.5.2). Both the decoder and full encoder-decoder are visualized in Figures 2.10, 2.11 respectively. The encoder and decoder can also exist as standalone models based on the desired use case. This is further discussed in Section 2.7 and 3.1.

2.5.5 Training Considerations

In practice, training transformers from scratch is difficult due to the complex interaction between the blocks' components. For example, the positioning of the normalization layer cause the gradients to shrink (Xiong et al., 2020) after the residual connection. Furthermore, the gradients for the queries and keys are significantly smaller than the values (Liu et al., 2020). The latter necessitates an adaptive algorithm combined with learning rate warmup to stabilize learning. *Gradient clipping* is another heuristic to ease complex model training by normalizing the gradient norm before the update step. This helps nurture exploding or diverging gradients in ill-conditioned loss landscapes.

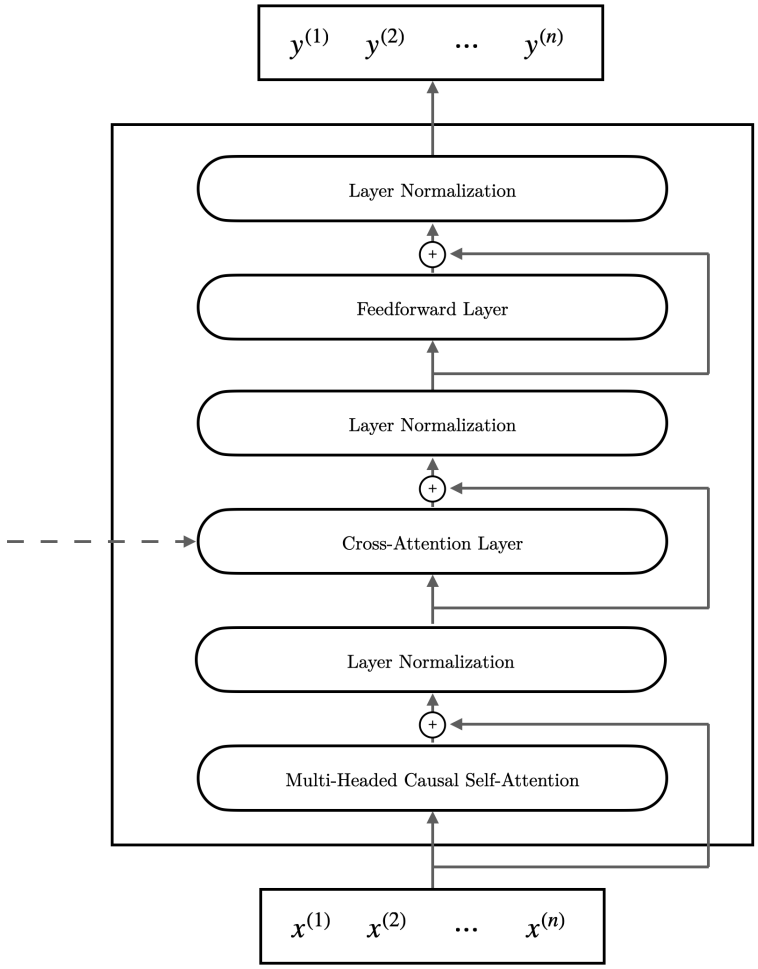


Figure 2.10: The decoder block for the Transformer encoder-decoder. Cross-attention uses the full bidirectional context whereas the causal attention preceding it masks the "future" context.

2.6 Language Models

Natural language processing (NLP) is the branch of statistical learning focusing on computational techniques to process, understand, and emit human language. We focus primarily on a specific class of models used to process and generate natural language. Goldberg (2016), Jurafsky and Martin (2022), and Eisenstein (2018) provide an exhaustive study on other applications in NLP.

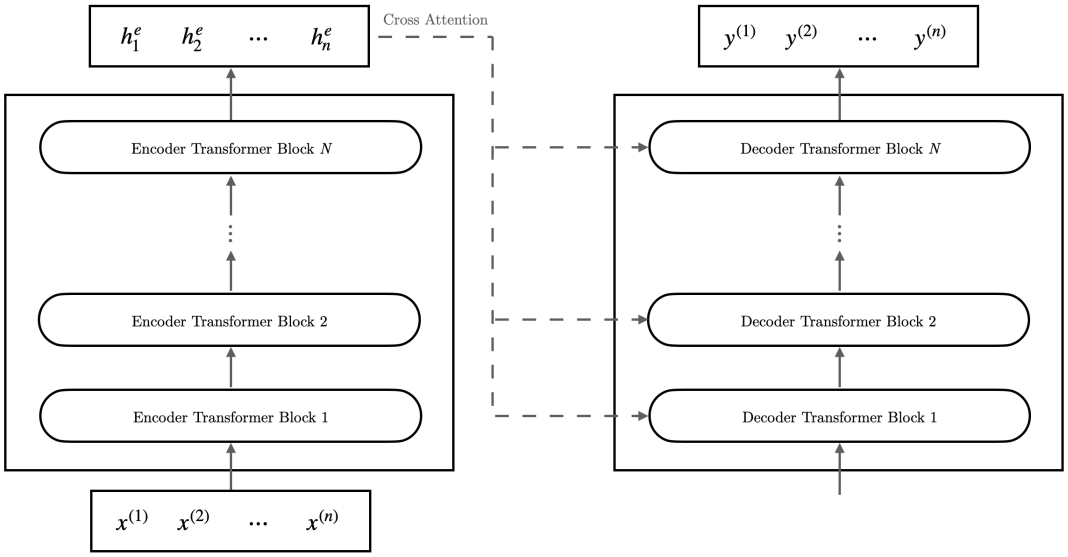


Figure 2.11: The Transformer as an encoder-decoder architecture. The input for the decoder is subject to the desired task.

2.6.1 N-Gram Language Models

Language models (LMs) are a class of *generative models* that assign probabilities to sequences of natural language *tokens*, or equivalently predict the next token in a sequence. Tokens can be either words, characters, bytes, or even subwords (Schuster and Nakajima, 2012; Kudo and Richardson, 2018; Kudo, 2018). For convenience, we will treat tokens as analogous to words and generalize in Section 2.6.4.

If $w_{1:t} = w_1 w_2 \cdots w_t$ denotes a sequence of t words, then LMs attempt to model $\mathbb{P}(w_{1:t}) = \mathbb{P}(w_1, \ldots, w_t)$. The probability can be decomposed using the *chain rule* as

$$\mathbb{P}(w_1, \ldots, w_t) = \mathbb{P}(w_1)\mathbb{P}(w_2|w_1)\mathbb{P}(w_3|w_{1:2}) \cdots \mathbb{P}(w_t|w_{1:t-1}) = \prod_{i=1}^{t} \mathbb{P}(w_i|w_{1:i-1})$$

Given a *training corpus*, the empirical estimate of $\mathbb{P}(w_i|w_{i-1})$ under maximum likelihood is the relative frequency of $w_{1:i}$ in the training corpus. Particularly,

$$\widehat{\mathbb{P}}(w_i|w_{1:i-1}) = \frac{\text{count}(w_{1:i})}{\sum_w \text{count}(w_{1:i-1}, w)} = \frac{\text{count}(w_{1:i})}{\text{count}(w_{1:i-1})}$$

where the denominator is simplified by the sum rule of probability.

In practice, calculating the estimates for extended histories (i.e. where $w_{1:i-1}$ is large) can be intractable. Instead, $\widehat{\mathbb{P}}(w_i|w_{i-1})$ is approximated using the *Markov assumption* where we assume that the conditional probability of w_t only depends on the last N words. Mathematically,

$$\mathbb{P}(w_i|w_{1:i-1}) \approx \mathbb{P}(w_i|w_{i-N-1:i-1})$$

where N is a hyperparameter and $w_{i-N-1:i-1}$ is the N-*gram* context (i.e. a contiguous sequence of N consecutive words). Putting this all together gives the MLE estimate

$$\widehat{\mathbb{P}}(w_i|w_{1:i-1}) \approx \widehat{\mathbb{P}}(w_i|w_{i-N-1:i-1}) = \frac{\text{count}(w_{i-N-1:i})}{\text{count}(w_{i-N-1:i-1})}$$

Statistical models leveraging n-grams formed the backbone of many early applications in NLP (Jelinek, 1976; Buck et al., 2014; Goodman, 2001; Jelinek, 1980; Baker, 1990; Chen and Goodman, 1996). There are major limitations to these models however, the first of which being any finite training corpus will yield zeros for valid n-grams not present in the data. This can be combatted by adding a *smoothing* term to redistribute the probability mass and assign unseen n-grams non-zero probabilities.

Similarly, n-gram models are prone to creating sparse inputs. For example, if V is the *vocabulary* (i.e. set of all unique words in the training set), then there are $|V|^N$ possible N-grams. Each n-gram is expressed as a *one-hot* vector of size $|V|^N$ where V is typically a large set. Furthermore, most of the anticipated $|V|^N$ n-grams aren't valid so associated vector components will always be zero.

Another issue is that these one-hot vectors are unable to encode semantic information. Intuitively, it is desirable to project words into a vector space where similar words appearing in similar context are closely related. For example, we expect the vector representations for *cat* and *dog* to be more similar than *cat* and *pineapple*.

2.6.2 Neural Language Models (NLM)

Neural language models are the family of LMs built from DNNs. These models overcome the limitations of using n-grams by projecting word representations into a *continuous* or *dense* space with dimension far less than $|V|^N$. This allows the model to encode each word as a distinct vector while simultaneously capturing information such as word similarity. Thus for words that are not in the training corpus, the model able to generalize and

conduct inference based on surrounding context. The dense representations from NLMs are called *word embeddings* and the projection layer is the *embedding layer*.

One of the earliest LMs was an MLP with one hidden layer proposed by Bengio et al. (2000). This model uses a *sliding window* to see the previous N words and a dedicated embedding matrix $\mathbf{E} \in \mathbb{R}^{d \times |V|}$. Each word is assigned a unique index $i = 1, \ldots, |V|$ corresponding to a column in the embedding matrix (i.e. a lookup table (Paszke et al., 2019; Abadi et al., 2016)).

The full forward propagation step is given by

1. *Select N embeddings from $\mathbf{E}$:*

   Given the previous N words, lookup their indices from the vocabulary and obtain their vectors from the embedding matrix (i.e. $\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_N}$). The embeddings are concatenated into $\mathbf{e} = \left[\mathbf{e}_{i_1}; \cdots; \mathbf{e}_{i_N}\right] \in \mathbb{R}^{Nd}$.

2. *Apply feedforward layer:*

   The hidden layer yields $\mathbf{h} = \text{ReLU}(\mathbf{We} + \mathbf{b})$.

3. *Compute word distribution:*

   The final layer is a softmax operation to obtain the distribution $\mathbb{P}[w_{t+1}|w_{t-N:t}]$. Each element i estimates $\mathbb{P}[w_{t+1} = i|w_{t-N:t}]$. The complete operation is $\widehat{\mathbf{y}} = \text{softmax}(\mathbf{Uh})$.

Transformers for Language Modelling

The autoregressive nature of the transformer decoder through its causal self-attention also makes it ideal as a *generative language model*. The decoder is trained directly to predict the next word at step t using all preceding tokens in the input rather than restricting to N-gram windows, i.e. $\mathbb{P}[w_t|w_{1:t-1}]$. The embedding layer is incorporated into the first decoder block and acts as the initial input before adding other encodings (i.e. positions).

During inference, the decoder generates tokens one at a time using its own outputs as preceding context. This makes it difficult to distribute the work along different compute. This is infeasible for training larger models so a workaround is to condition on the *ground truth*. This is known as *teacher forcing* since the true labels are *force fed* into the model as input at each step. Additional language models are discussed in Section 2.7 and 3.1.

2.6.3 Sampling Outputs

The process of sampling the next word from the decoders' probability distribution is known as *decoding*. The simplest approach to greedily choose the most likely word at each step, however this doesn't guarantee that the final sequence is the more probable. An exhaustive search is intractable as it requires $|V|^{\tau}$ permutations.

Beam search is a compromise between the two heuristics by maintaining k candidate outputs at each step, called the *hypotheses*. At each step, each candidate is expanded to include all V possible values where we again choose the top k and repeat. To prevent bias towards shorter strings, beam search incorporates length normalization into its scores. It is also possible to incorporate n-gram penalties (Paulus et al., 2017; Klein et al., 2017) to prevent repetitive generation by forcing the next-token probability to zero where applicable. However, the "best" n will depend on domain context and is difficult to optimize in general. Other variants of beam search aim to diversify the hypotheses (Vijayakumar et al., 2016; Kulikov et al., 2018).

Holtzman et al. (2019) showed that language generated by humans does not follow the probabilistic next-word model established for LMs. It is possible to randomly sample tokens according to the model distribution, however in practice there is a long tail of unlikely words that collectively make up significant mass. Fan et al. (2018) proposed *top-K sampling* in which tokens are only sampled from the K most likely hypotheses. This can still be problematic if there are a limited number of high-probability tokens, so Holtzman et al. (2019) developed *nucleus sampling* where the tokens are sampled from a fixed proportion of the total probability mass instead.

2.6.4 Tokenization and Vocabulary

Any text can be decomposed into a sequence of atomic units called *tokens*. Up to this point, we assumed that tokens and words were interchangeable, however they can also be characters, bytes, or subwords. The process of converting text into tokens is called *tokenization*. The *vocabulary* is then a mapping between tokens and unique indices (this process may be called *encoding*).

In most cases, special indices in the vocabulary are reserved for specific tokens. For example, the *unknown token* is reserved as a value for any token that does not appear in the vocabulary. Similarly, tokens can be used to indicate the beginning or end of a sequence. A token can also be used specifically to "pad" or inflate a sequence to some specified length.

The simplest form of tokenizing English text is segmenting based on whitespace and punctuation. However, this may not be applicable to other languages where words are not separated by boundaries or characters have higher individual significance. It is more common to tokenize the text based on characters in these cases.

2.6.5 Subword Tokenization

An issue with word-level tokenizers is that they are prone to encountering *unknown tokens* and cannot capture syntactical variations such as suffixes. A solution is to use word fragments or *subwords* to be tokens allowing for unknown words to be further decomposed into known tokens. Subword tokenization schemes have two parts, a *token learner* and *token segmenter/parser*. The learner is responsible for taking a training corpus and deriving a vocabulary and set of segmentation rules. The segmenter then uses these rules to convert any text into a sequence of associated tokens.

Byte-pair encoding (BPE) (Sennrich et al., 2016) and *WordPiece* (Schuster and Nakajima, 2012) are two popular tokenizers that greedily merge pairs of tokens from an initial set until the specified vocabulary size is reached. BPE starts with characters and punctuation (or bytes (Radford and Narasimhan, 2018)) and merge based on adjacent frequency. WordPiece starts with characters in the training corpus and merges based on likelihood (i.e. s_1, s_2 are merged if the empirical frequencies of $s_1 s_2$ divided by s_1 and s_2 is the greatest among all pairs).

The *unigram* language model (Kudo, 2018) works in the opposite direction by starting with a large vocabulary (generated by another algorithm) and removing tokens. Specifically, tokens are pruned if doing so results in the smallest increase in likelihood are pruned based on minimal increase in the NLL loss of a language model.

All of the algorithms described so far assume that the input is white-separated first. *SentencePiece* (Kudo and Richardson, 2018) treats the entire input as a raw input stream thus including whitespace as a valid character when applying BPE or Unigram. The whitespace character is typically denoted with another character such as "_."

2.7 Transfer Learning

Many tasks with low-amounts of data benefit from sharing high-level similarity to data-rich tasks. For example, many downstream language processing tasks share general language

semantics and characteristics from a larger corpus (assuming the language is the same). This suggests that first training a model over a large dataset to capture general language intricacies before continually training on a small downstream dataset would lead to better performance (Howard and Ruder, 2018). This can be viewed as developing a good representation of the data beforehand or initializing the parameters to a space where a good solution is more likely.

This is the premise of *transfer learning*. The first stage of training a model with parameters $\boldsymbol{\theta}$ on a large *source dataset* $\mathcal{D}_s$ is called *pre-training*. The second phase of continually training the model on a *target dataset* $\mathcal{D}_t$ is called *fine-tuning*. Typically the model is modified to match the target distribution by replacing or extending the last layer(s). For example, if the output distribution for a pretrained DNN is $\mathbb{P}[y|\mathbf{x}, \boldsymbol{\theta}] = \text{softmax}(\mathbf{W}_1\mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_1) + \mathbf{b}_1)$, the new *head* becomes $\mathbb{P}[y|\mathbf{x}, \boldsymbol{\theta}_t] = \text{softmax}(\mathbf{W}_2\mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_1) + \mathbf{b}_2)$ where $\boldsymbol{\theta}_1$ is the pretrained model's parameters up to the final layer. To prevent *catastrophic forgetting* (McCloskey and Cohen, 1989), the learning rate is usually configured as a low value so that the model weights don't deviate significantly.

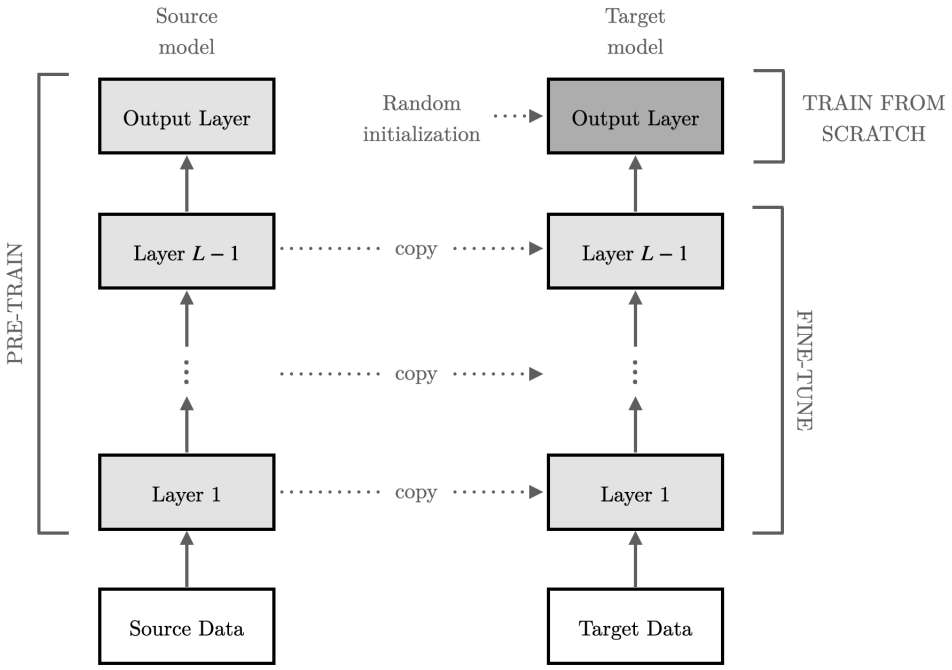


Figure 2.12: The transfer learning paradigm for a multilayered model (i.e. deep neural network). This figure is based off the one in Murphy (2022).

2.7.1 Pre-training

The pre-training task may be supervised or unsupervised as the goal is for the model is able to learn sufficient context for downstream generalization. This is not rigourously defined, however it was shown that pre-training task similarity contributes significantly to downstream performance (Zhang, Zhao, Saleh and Liu, 2019) (i.e. no-free lunch theorem).

Supervised pre-training is more common in computer vision applications (Kolesnikov et al., 2019*b*) because of the availability of large labelled datasets (Deng et al., 2009). However, due to the accessibility of unlabelled data (especially text) *self-supervised pre-training* is more popular in recent work. These techniques are coined "self-supervised" as the pre-training task is discriminative, however the labels are assigned using an algorithm as opposed to human-annotations. One such example was the next-word prediction discussed to train decoders in Section 2.6.2.

2.7.2 Fine-tuning

Standalone transformer encoder models are typically used to transform discrete text inputs into contextualized token representations (Peters et al., 2017, 2018) that can generalize to a variety of downstream tasks (i.e. sequence/token classification, question answering, etc.). Thus these models are trained in a self-supervised manner over a large corpus so that they can exploited for transfer learning. For example, they may be optimized to reconstruct inputs where tokens are randomly masked or replaced (*Masked Language Modelling* or MLM) (Devlin et al., 2019; Liu et al., 2019; Joshi et al., 2020). To fine-tune the model, the appropriate head is simply attached to the final encoder representation. Because the goal of the encoder is to learn representations, the full self-attention mechanism is used to incorporate *bidirectional context* (Devlin et al., 2019).

Early transfer learning for NLP relied on finetuning autoregressive language models for discriminative tasks (Howard and Ruder, 2018). However, these are largely replaced with encoders and contextualized representations. Decoders are primarily dedicated to generative modelling where they demonstrated recent success in *few-shot generalization* through *prompting*.

Chapter 3

Literature Review

This chapter provides an overview of two areas of research relevant to our work. The first section discusses recent advancements and applications revolving around the transformer architecture (introduced in Section 2.5). The second section reviews prior work in data-to-text generation and sports game summarization. Specifically, prior work in improving data purity in existing sports game summarization benchmarks directly motivate the problem of interest in this thesis.

3.1 Modern Transformers

The original transformer (Vaswani et al., 2017) was orientated towards translations tasks, but was quickly adopted to various applications in NLP, computer vision, and other modalities. It is also common for standalone encoders or decoders to be trained for specific use cases. Transformers have largely found success due to ease of scale and parallelism, as well as superior performance in many objectives when pre-trained over large amounts of data. NLG is one of the many areas where transformers surpassed early state-of-the-art benchmarks and thus will be the model of choice when we benchmark our own DTG datasets in Chapter 4.

This section provides an overview of recent advances under the transformer architecture, such as efficient attention mechanisms, encoder-decoders, incorporating to computer vision, and generative language models.

3.1.1 Sparse Attention Mechanisms

Recall from Section 2.5.1 that self-attention has $O(N^2)$ compute and memory complexity. This poses a practical limit on the length of the input sequence for tasks such as summarization. A common strategy is to leverage a *sparse attention matrix*, though it is also possible to project the matrix to a smaller size (Liu et al., 2018; Wang et al., 2020) or replace the softmax operation with an efficient equivalent (Katharopoulos et al., 2020; Choromanski et al., 2020).

One approach is to restrict the range of keys to some local neighbourhood of the query (Liu et al., 2018; Ainslie et al., 2020; Beltagy et al., 2020), aptly named *local*, *sliding window* or *convolutional* attention. A related approach is to group tokens into blocks such that tokens within a block only attend to one another (Phang et al., 2022; Liu, Lin, Cao, Hu, Wei, Zhang, Lin and Guo, 2021). Due to their close relationship to convolutional layers (LeCun et al., 1989), local attention can also implement similar extensions to expand the network's *receptive field* such as dilation rates. However, local attention alone severely limits the predictive power for transformers for more complex tasks. To counteract this, local layers are either alternated with full-attention (Brown et al., 2020) or combined with dedicated global (Beltagy et al., 2020; Ainslie et al., 2020; Guo et al., 2022; Phang et al., 2022) and random (Zaheer et al., 2020) tokens.

3.1.2 Encoder-Decoders

Transformer encoder-decoders are widely adopted in natural language generation (NLG) for sequence-to-sequence tasks such as machine translation, text summarization, or dialogue generation (Freitas et al., 2020). The encoder uses bidirectional attention to fully encode the input into a *contextual representation*. The decoder then uses causal attention to auto-regressively generate the output sequence while using cross-attention to attend to the encoder representation. Most of the state-of-the-art models in this class generally differ in pre-training objectives and downstream purpose. In terms of architecture, the main differences are constrained to activation functions, modified blocks, or using an efficient self-attention described earlier.

BART (Bidirectional and Auto-regressive Transformer) (Lewis et al., 2019) is an encoder-decoder trained on a self-supervised denoising objective. Specifically, the model was pre-trained to reconstruct corrupted inputs where continguous spans are masked (*text infilling*) and the sentences are randomly shuffled (*sentence permutation*). This model is best suited for summarization tasks however it is possible to extend the encoder or decoder to finetune

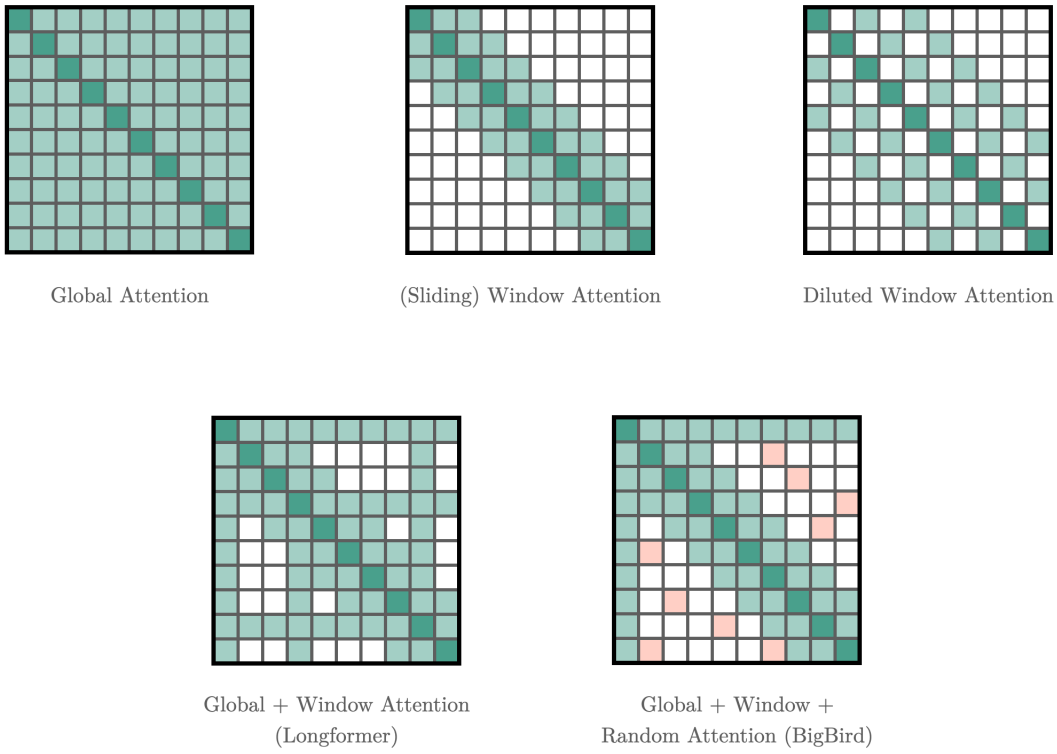


Figure 3.1: Visualization of different sparse attention matrices as described in Section 3.1.1.

on more. The *LED* (Longformer Encoder-Decoder) (Beltagy et al., 2020) an adaptation of BART which replaces the encoders' self-attention with a sparse variant (dilation + local + global token at the first position).

Similarly, *PEGASUS* (Pre-training with Extracted Gap-sentences for Abstractive Summarization) (Zhang, Zhao, Saleh and Liu, 2019) was trained on a denoising objective where entire sentences are masked based on their significance in relation to the input (*Gap Sentences Generation* or GSG). Strategically masking the inputs in this manner loosely resembles abstract summarization. Altogether, the encoder is trained with masked language modelling while the decoder simultaneously generates only the masked sentences. *PEGASUS-X* (Phang et al., 2022) and *BigBird* (Zaheer et al., 2020) extend the model to incorporate sparse self-attention independently. There are also models that use the GSG objective for long-sequence summarization (Guo et al., 2022).

In contrast to the previous examples, *T5* (Text-to-Text Transfer Transformer) (Raffel

et al., 2019) was pre-trained simultaneously on a novel reconstruction objective and a plateau of other supervised tasks to produce a multi-faceted model with potential zero-shot generalization. This is done in a strictly "text-to-text" manner where the inputs and outputs are text sequences regardless of the task. This provides the benefit of training everything under a single cross-entropy loss.

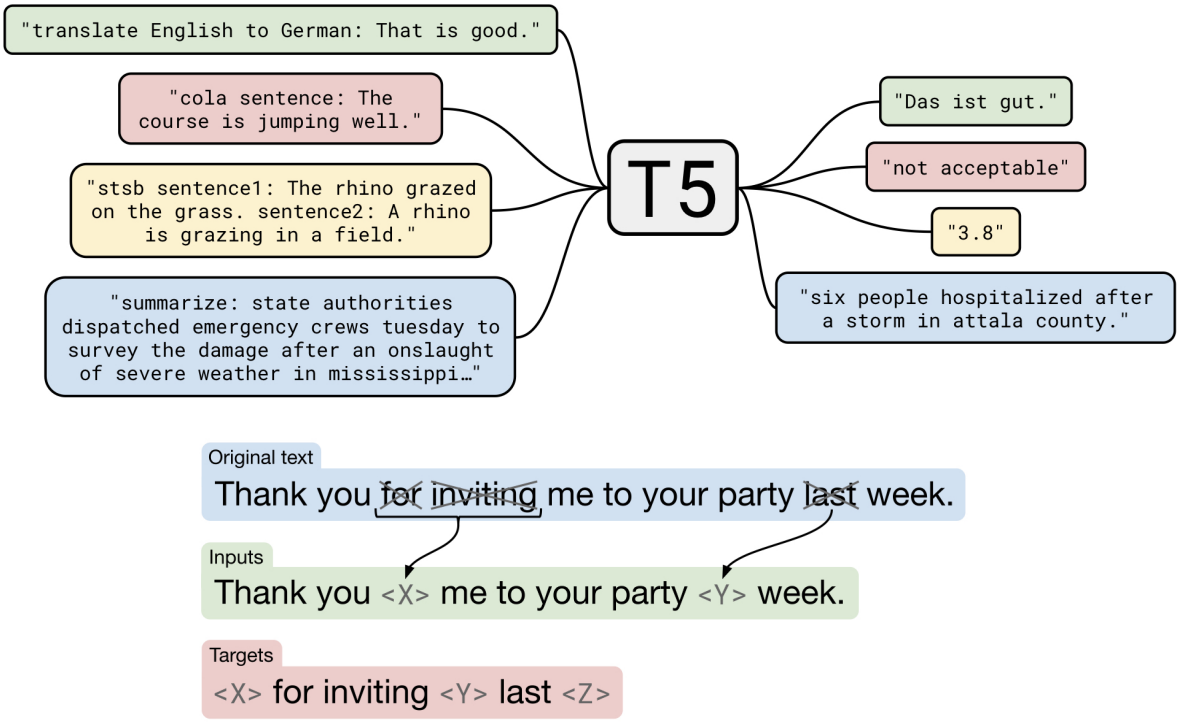


Figure 3.2: The text-to-text framework of T5, images from Raffel et al. (2019). (*Top*) The tasks are specified as part of the input and the model output is always parsed as human-readable text. (*Bottom*) An example of the unsupervised objective for T5, note the model groups predictions with the sentinel tokens.

To differentiate between tasks, the inputs are fitted with a prefix specifying what the model should output (see Figure 3.2). The unsupervised objective is similar to BART where the model has to reconstruct masked contiguous spans via unique sentinel tokens. However the model only generates the masked portions instead of the full corrected sequence, thus reducing the loss complexity. T5 also adopts a relative position scheme where scalar embeddings are added to the corresponding attention logits. Each embedding maps to

an increasing *range* (or bucket) of offsets where all tokens beyond the full window are assigned the same embedding.

In a similar manner, *FLAN* (Finetuning Language Models) (Wei, Bosma, Zhao, Guu, Yu, Lester, Du, Dai and Le, 2022; Chung et al., 2022) is a fine-tuning procedure extending multitask learning by rephrasing data inputs as explicit instructions. This model-agnostic objective has demonstrated promising results on several architectures (Raffel et al., 2019; Chowdhery et al., 2022; Tay et al., 2022) in zero and few-shot generalization on a variety of unseen tasks. For example, including *chain-of-thought prompting* (Wei, Wang, Schuurmans, Bosma, hsin Chi, Le and Zhou, 2022) improves a language model's general ability to reason by having it generate its intermediate steps to influence conciseness.

3.1.3 Generative Language Models

Many language models built from transformer decoders, as described in Section 2.6.2, have become the forefront of modern NLP due to their unprecedented success in few-shot generalization. This is particularly noteworthy as significant advancements in SOTA benchmarks come almost exclusively from scaling and sufficient training (Hoffmann et al., 2022) over various data sources (though this comes with the disclaimer of possible data contamination based on training corpus).

The *GPT* family (Generative Pre-training Transformer) (Radford and Narasimhan, 2018; Radford et al., 2019; Brown et al., 2020; OpenAI, 2023; Black et al., 2022; Ouyang et al., 2022) are decoders trained with next-word prediction. GPT-3 pushed the scale of the canonical LMs at 175 billion parameters and was trained on *The Pile* (Gao et al., 2021). Due to the scale and diversity of training data, GPT-3 demonstrated an ability to generate novel text. Specifically, the output can be somewhat influenced by conditioning on an *initial prompt*. In some cases the model is able to conduct *few-shot task transfer* when prompted with several examples. This is an example of *in-context learning*.

More recently, *InstructGPT* (the precursor to *ChatGPT*) (Ouyang et al., 2022) uses *RLHF* (reinforcement learning from human feedback) (Christiano et al., 2017) to fine tune GPT-3 over outputs that are "aligned" with human intent using a separate ranking model trained over supervised data. The resulting model has shown to generate more desirable outputs and follow instructions better at a significantly smaller scale (1.3B v.s. 175B).

The breakthrough with GPT-3 started a series of *large language models* (LLMs) that placed an emphasis on scaling. These include *Megatron-Turing NLG* (Smith et al., 2022), *GLaM* (Du et al., 2022), *Gopher* (Rae et al., 2021), *Chinchilla* (Hoffmann et al., 2022), *PaLM* (Chowdhery et al., 2022), *OPT* (Zhang et al., 2022), *LaMDa* (Thoppilan et al.,

2022), and most recently (at the time of writing) *LLaMA* (Touvron et al., 2023). Almost all of the improvement can be attributed to scale, efficient training procedures, and exploiting large diverse datasets.

3.1.4 Vision Transformers

Transformers were also adopted into *computer vision* by modifying the architecture to process image data. Unlike text, images can be processed directly as three-dimensional tensors of size $H \times W \times C$. Specifically, H is the height, W is the width, and C is the number of channels (i.e. RGB is $C = 3$ for red, green and blue). Each element at position (i, j, k) is the *pixel intensity* at channel k. In practice, the pixel intensities range from $[0, 255]$ and are usually standardized to $[0, 1]$ or $[-1, 1]$. This is normally the only preprocessing done, however many pipelines also re-scale or crop the images to some fixed size.

Prior to this, computer vision was dominated by *convolutional neural networks* (CNNs) (LeCun et al., 1989; He et al., 2015*a*; Krizhevsky et al., 2012). These networks were designed to specifically handle inputs exhibiting a grid-like topology. Furthermore CNNs uniquely inherit additional properties such as locality and spatial translation invariance making them ideal for these types of inputs (i.e. good *inductive biases*). Transformers process images as an arbitrary sequence thus necessitating additional learning to surmount this disadvantage. Despite this, transformers have largely eclipsed the CNN benchmarks on various vision tasks (Kolesnikov et al., 2019*a*; Mahajan et al., 2018; Xie et al., 2019) due to their scale and accessibility of large pre-training datasets (Sun et al., 2017; Deng et al., 2009; Schuhmann et al., 2021).

The *ViT* (Vision Transformer) (Dosovitskiy et al., 2020) is an encoder which processes images by dividing them into 16×16 non-overlapping patches that are further down-projected and added to learned position embeddings. A dedicated `[CLS]` classification token (Devlin et al., 2019) is prepended to the input sequence and the model was pre-trained on the supervised ImageNet dataset (Deng et al., 2009). *BEiT* (Bao et al., 2021) is an updated variant that adopts the MLM pre-training task by reconstructing masked image patches.

Swin (Hierarchical ViT using Shifted Windows) (Liu, Lin, Cao, Hu, Wei, Zhang, Lin and Guo, 2021) is a modified ViT that processes larger images efficiently by *downsampling* the resolution and increasing the number of channels (reminiscent to CNN backbones). This encoder divides the images into patches that are then grouped into non-overlapping windows in which self-attention is applied independently. To retain predictive power, each

alternating transformer block *shifts* the windows so different subsets of patches can interact. The resolution is periodically reduced by merging neighbourhoods of non-overlapping patches and projecting them to increase the number of channels. Swin adopts relative position bias and uses bi-cubic interpolation to scale to different window sizes (Dosovitskiy et al., 2020; Touvron et al., 2020).

Swin V2 (Liu, Hu, Lin, Yao, Xie, Wei, Ning, Cao, Zhang, Dong, Wei and Guo, 2021) updated the architecture to stabilize learning at larger scales and improve performance for high-resolution transfer. To improve training, the attention scores are replaced with scaled cosine similarities and post-residual normalization. The relative position bias is implemented as a continuous function with log-scale positions, significantly reducing the extrapolation step size at larger resolutions.

The transformer architecture has also been modified to handle other canonical computer vision tasks such as image classification (Touvron et al., 2020), object detection (Carion et al., 2020; Zhu et al., 2020; Fang et al., 2021), semantic segmentation (Ye et al., 2019; Xie et al., 2021; Gu et al., 2021), and image generation (Chen, Wang, Guo, Xu, Deng, Liu, Ma, Xu, Xu and Gao, 2020; Nash et al., 2021).

3.1.5 Multimodal Transformers

Accommodating the transformer for various input formats allows for unifying different data types into a single end-to-end model with ease. These types of models are classified as *multimodal* to distinguish their ability to encode or generate different modalities. For example, the *vision encoder-decoder* (Li et al., 2021; Wang et al., 2022; Kim et al., 2022) combines a vision encoder with an autoregressive language decoder for image-to-text tasks such as image captioning or optical character recognition (OCR). It is also possible to interleave multiple modalities within a single input sequence for simultaneous processing.

The *LayoutLM* family (Xu et al., 2019, 2020; Huang et al., 2022; Xu et al., 2021) are foundation document-understanding encoders fitted to combine textual and visual cues in their input representations. As the text, image patch embeddings, and positions have to align these models are typically reserved for document processing with an additional OCR method. These encoders are pre-trained on reconstruction tasks revolving around masking tokens and portions of the image.

DONUT (OCR-free Document Understanding Transformer) (Kim et al., 2022) is an end-to-end vision encoder-decoder that can comprehend document images without explicitly providing the aligned text. It was pre-trained on a pseudo-OCR objective on augmented

images (Yim et al., 2021; Lewis et al., 2006) at high resolutions. Similarly, *GiT* (Generative Image Transformer) Wang et al. (2022) is a general-purpose vision encoder-decoder that was trained over a large quantity of image-text pairs.

There is also an ongoing surge (at the time of writing) of foundational multimodal language models (Driess et al., 2023; Huang et al., 2023; OpenAI, 2023) that were trained on unified input types to generalize over many unseen generative tasks. This is the natural extension of decoder-based LLMs by incorporating additional modalities to aid with conciseness on tasks benefiting from multimodal prompts.

3.1.6 Evaluating Text Generations

The cross-entropy acts as a *pseudo-loss* in order to train the model, however the metric of interest is usually different (i.e. accuracy, precision, recall, etc.). Most metrics that are used to evaluate language fluency require both *predicted* and *reference* texts and focus on shared n-grams. For example, the *Bilingual Evaluation Understudy* (BLEU) (Papineni et al., 2002) is a measure of similarity between $[0, 1]$ calculated by counting n-gram occurrences between the two texts up to $n = 4$. They further include a *brevity penalty* to penalize shorter predictions. Despite its popularity, it was shown that BLEU correlates poorly with human judgement for NLG (Novikova, Dusek, Curry and Rieser, 2017).

Similarly, the *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE) score (Lin and Hovy, 2003; Lin, 2004; Lin and Och, 2004) is an extension of BLEU that emphasizes n-gram recall. There are several notable variants of the metric such as ROUGE-1 which computes the overlap proportion of *unigrams*. Similarly, ROUGE-2 computes the overlap proportion of *bigrams*. ROUGE-L (Lin and Och, 2004) looks at the longest common subsequence between the reference and candidate texts.

The *Metric for Evaluation of Translation with Explicit ORdering* (METEOR) (Banerjee and Lavie, 2005) is another variant of BLEU that takes the harmonic mean between unigram precision and recall and accounts for stemming and synonyms (Miller, 1992). Thus, METEOR first looks to construct an *alignment* between the candidate and reference texts through exact, stem, or synonym matches.

The contextualized token representations of LLM encoders (see Section 2.6.2) can also be utilized to construct text similarity measures. Unlike with the n-gram metrics the vectors' encoded semantic information are also taken into account. COMET (Rei et al., 2020) and BLEURT (Sellam et al., 2020) trained or finetuned models on datasets with human-annotated similarity scores between two text segments. Likewise, BERTScore (Zhang et al., 2020) computes pairwise cosine similarity between all the token embeddings between both

texts and greedily match the most similar token to compute the precision and recall. As these methods still involve large transformers, concerns associated with compute and inference bottlenecks are still valid.

3.2 Data-to-Text Generation

Data-to-Text Generation (DTG), following the seminal work of Reiter and Dale (1997), is concerned with the automation of translating structured data instances (i.e. tables, graphs, etc.) into coherent narratives. Unlike other tasks from NLG, the input is not exclusively a linguistic representation of information and can incorporate different data structures within a single input. Furthermore this objective poses additional challenges compared to canonical sequence transduction as the generator needs to determine *what to say* (i.e. selecting the appropriate subset of input data to discuss) on top of *how to say it*. This topic is adopted in various commercial and proprietary frameworks and garners considerable applicability in different domains (Portet et al., 2007; Pauws et al., 2018; Anselma and Mazzei, 2018; Murakami et al., 2017; Aoki et al., 2018; Juraska et al., 2019; Braun et al., 2018). This section focuses on earlier work revolving around neural networks and focuses specifically on the applicability of LLMs and the sports game summarization case. For a more general overview of neural network solutions on various DTG datasets, refer to Sharma et al. (2022).

3.2.1 Datasets

The problem of DTG is diverse in the sense that the input representations can come in many forms such as *meaning representations*, *graphs*, *tabular structures*, etc. Furthermore the assigned narratives can vary in complexity and can pose additional challenges themselves. Due to similar example structures, DTG is closely aligned to tasks such as machine translation, summarization, and even domain-specific tasks like image captioning and question answering. Because of this a lot of the techniques and models from related areas tend to generalize well when adopted to DTG (i.e. neural encoder-decoders).

The *WebNLG* dataset (Gardent et al., 2017) is a collection of graph-text pairs spanning various domains. The graph nodes are encoded as *Resource Description Format* (RDF) triplets and the text is parsed in both English and Russian making it one of the few non-anglo centric DTG datasets. *AGENDA* (Koncel-Kedziorski et al., 2019) is another graph-to-text dataset that pairs knowledge graphs extracted from AI conference proceedings to

article abstracts. *DART* (Radev et al., 2020) is a cross-domain dataset formed by merging the meaning representation dataset *E2E* (Novikova, Dusek and Rieser, 2017) and WebNLG.

The *WikiBio* dataset (Lebret et al., 2016) is a large scale table-to-text dataset with pairs of Wikipedia info-boxes and the first paragraph of the associated article. The vocabulary size and number of examples far exceeded the existing pioneering table-to-text datasets (Chen and Mooney, 2008; Liang et al., 2009). The *Rotowire* (Wiseman et al., 2017) and *MLB* (Puduppully et al., 2019*a*) datasets posed new challenges as they paired data records over multiple tables to detailed narratives summarizing in-game events. The average reference length and number of data records are substantially larger than contemporary table-to-text datasets, even at the time of writing.

ToTTo (Parikh et al., 2020) is an open-domain *controlled* generation task that requires generating short narratives from tables with pre-selected/highlighted cells. *SciGen* (Moosavi et al., 2021) incorporates implicit arithmetic reasoning over table entries into its narratives. *WikiTableT* (Chen, Wiseman and Gimpel, 2020) is another dataset with 1.5 million examples that pairs Wikipedia descriptions to multiple tables and metadata to produce a large-scale benchmark inspired by the high-dimensional challenges of Rotowire.

There are also datasets with niche input representations (though graphs and tables are typically the most common). For example, there are two datasets aptly named *chart-to-text* (Obeid and Hoque, 2020; Kanthara et al., 2022) that pair visual charts and graphs to narrative descriptors. Similarly, (Sharma et al., 2021) assigns narratives to time-series graphs.

3.2.2 Applicability of LLMs

Kale (2020) demonstrated that the text-to-text pretrained T5 was able to outperform many pipelined neural architectures on several DTG datasets. The input data instances were simply cast as flat strings (called *linearization*) and passed into the different variants of T5. Their experiments found that the "knowledge" possessed by these models at increased capacities played a prominent role for out-of-domain generalization. For instance, since T5 was trained on a large unsupervised web corpus, the larger variants likely have domain-specific word distributions incorporated into its weights.

Further expanding this revelation, Yermakov et al. (2021) applied additional generative LLMs in the biomedical domain where more complicated generation is required (similar to sports summarization, discussed in Section 3.2.3). Keymanesh et al. (2022) and Mehta et al. (2022) study the generalization properties of using these same LLMs in DTG. In the

extreme case, Kasner and Dusek (2022), Chang et al. (2021), and Su et al. (2021) propose methods and pipelines for few and zero-shot DTG with LLMs.

3.2.3 Sports Game Summarization

Sports game summarization is a domain-specific NLG objective concerned with producing coherent summaries of sports games. This was originally approached as its own area of research (Robin, 1994; Tanaka-Ishii et al., 1998; Barzilay and Lapata, 2005) but is now largely incorporated as a special case of DTG. *Rotowire* and *MLB* are such datasets and involve generating multi-sentence summaries by selecting from a substantial number of records encompassing numerous tables. A related line of research is evaluating the "truthfulness" of the generated statistics and relations on top of language fluency (further discussed in Section 4.3). An example from the ROTOWIRE dataset is given in Figure 3.3.

TEAM	WIN	LOSS	PTS	FG_PCT	RB	AS ...
Heat	11	12	103	49	47	27
Hawks	7	15	95	43	33	20

PLAYER	AS	RB	PT	FG	FGA	CITY ...
Tyler Johnson	5	2	27	8	16	Miami
Dwight Howard	4	17	23	9	11	Atlanta
Paul Millsap	2	9	21	8	12	Atlanta
Goran Dragic	4	2	21	8	17	Miami
Wayne Ellington	2	3	19	7	15	Miami
Dennis Schroder	7	4	17	8	15	Atlanta
Rodney McGruder	5	5	11	3	8	Miami
Thabo Sefolosha	5	5	10	5	11	Atlanta
Kyle Korver	5	3	9	3	9	Atlanta
...						

The Atlanta Hawks defeated the Miami Heat , 103 - 95 , at Philips Arena on Wednesday . Atlanta was in desperate need of a win and they were able to take care of a shorthanded Miami team here . Defense was key for the Hawks , as they held the Heat to 42 percent shooting and forced them to commit 16 turnovers . Atlanta also dominated in the paint , winning the rebounding battle , 47 - 34 , and outscoring them in the paint 58 - 26.The Hawks shot 49 percent from the field and assisted on 27 of their 43 made baskets . This was a near wire - to - wire win for the Hawks , as Miami held just one lead in the first five minutes . Miami (7 - 15) are as beat - up as anyone right now and it 's taking a toll on the heavily used starters . Hassan Whiteside really struggled in this game , as he amassed eight points , 12 rebounds and one blocks on 4 - of - 12 shooting ...

Figure 3.3: An example from ROTOWIRE. Each feature is a collection of statistical records in the form of a boxscore table. The target is the associated human-generated summary describing the events of the game. Image from Wiseman et al. (2017).

The inaugural work of Wiseman et al. (2017) adopted an end-to-end LSTM encoder-decoder with attention (Luong, Pham and Manning, 2015) and embedded the records

directly as input vectors. To improve the decoder's precision they also fitted a *copy mechanism* (Gu et al., 2016; Yang et al., 2016; Gulcehre et al., 2016) that was tasked with copying record values directly from the input and modelling when to do so. To automatically evaluate the correctness of the generation, the authors propose three different criteria:

- *Content Selection (CS):* compute the precision and recall between the unique relations extracted from $\hat{y}_{1:T_1}$ and $y_{1:T_2}$.
- *Relation Generation (RG):* compute the precision and number of unique relations r extracted from $\hat{y}_{1:T_1}$ that appears in the database $\mathbf{s}$.
- *Content Ordering (CO):* compute the normalized Damerau-Levenshtein distance (Brill and Moore, 2000) between the *sequence of records* extracted from $\hat{y}_{1:T_1}$ and $y_{1:T_2}$.

Wiseman et al. (2017) built entity extraction models to collect relations from the summaries where a noisy dataset was created using string heuristics to find all pairs of entities and numeric values in each sentence in the gold summaries. These pairs are then mapped back to the input table to determine the relation.

Puduppully et al. (2019*a*) explicitly model the *content selection* and *planning* phases as separate components. They leveraged a bidirectional LSTM *pointer network* (Schuster and Paliwal, 1997; Vinyals et al., 2015) that returns an ordered sequence of indices (i.e. pointers) in the original sequence representing the selected records or the *content plan*. The decoder then generates the summaries by conditioning on this content plan. Gong et al. (2019) model the selection phase with a transformer encoder that predicts if the record is used in the summary. The encoder was trained by alternating between the classification and LM objective. Similarly, Narayan et al. (2020) proposed a stepwise approach to creating the content plans using pre-trained structured transformers - *HiBERT* (Zhang, Wei and Zhou, 2019) and *ETC* (Ainslie et al., 2020). Puduppully et al. (2019*b*) further extend the encoder by introducing *entity-specific* representations as opposed to treating them as regular tokens.

Puduppully and Lapata (2021) propose abstracting beyond content plans to *macro plans* which are sequences of *paragraph plans* that themselves are sequences of entities and their associated records. The pointer network is then tasked with selecting the paragraph plans whose pointers form the macro plan. Puduppully et al. (2022) update the plan encoder to increment by paragraph and condition each plan on both the previous plans as well as the generated paragraph(s).

Outside of explicit plan modelling, Rebuffel et al. (2019) revise the encoder to fit the *hierarchical* structure of the data tables and retain the record order and other distinctions

between entities (i.e. team v.s. player). The *low-level* encoder produces a single representation for each entity which the *high-level* encoder aggregates a dynamic context over all entities at each decoder step. On the other end, Choi et al. (2021) introduce two different copy mechanisms to improve the precision at the decoding stage.

Ethan Joseph and Si (2021) note the poor generation quality of existing approaches and opt to use LLMs as the base encoder-decoder. Unlike Kale (2020), they include additional tokens to identify different record or relation types (i.e. `<TM-PTS>`, `<TM-REB>`, etc.). Furthermore, they included an auxiliary loss to encourage generating common trigrams and imposed length and n-gram penalities (Paulus et al., 2018). They also use a *fact-check* module which predicts the relation of the sentence and subsequently replace the value.

Despite all of the work in the literature pertaining to this dataset, the summaries themselves suffer from low data fidelity. For example, many of the statements incorporate additional domain expertise and background knowledge consolidated from sources outside of the input records. Wang (2019) observed that approximately 40% of the game summary contents couldn't be mapped to any input in the boxscore records (57% of which are still game-specific items). This encourages the models to *hallucinate* facts that consequently inflate language fluency scores without being penalized or addressed in any other correctness metrics. The author proposes purifying the summary by removing all sentences that couldn't be mapped back to a record in the data table. Chapter 4 further explores the data fidelity problem for ROTOWIRE and proposes solutions through a domain-driven perspective.

Chapter 4

Large Data-to-Text Generation

As mentioned in Section 3.2.3, Wang (2019) addresses the data fidelity issue within the ROTOWIRE dataset. They propose primarily to remove sentences in the summary that can not be mapped to the collection of input records. However, this poses two issues where (1) the final gold summary may no longer be coherent; and (2) the model fit may not be suitable outside of a research setting as the retained information may only be trivial. Instead, we propose working in the other direction and take advantage of the observation that most of the ungrounded phrases are still game-specific. That is, we procure a larger collection of input records with supplementary information more aligned with what appears in the summaries. Doing so introduces a more complex layer on top of the existing DTG problem as the number of records becomes astronomical and necessitates the need for efficient models as well.

This chapter first formalizes the idea of *large data-to-text generation* which we coin as the specific case of DTG with a substantial number of input records distributed across various representations. We then expand on the automatic evaluation models of Wiseman et al. (2017) and propose a new pipeline using generative models and data augmentation over an extended number of entities and relations. We then introduce the supplementary input records and demonstrate their impact on model performance against a cleaned version of the original ROTOWIRE dataset. To do so, we would need to use the sparse attention LLMs discussed in Section 3.1.1 and validate their effectiveness on non-linguistic inputs. On top of the LLMs we also propose a new way of tackling the sports summarization problem itself through image inputs with embedded data structures. Throughout the chapter we introduce novel datasets and analyze the impact of vision encoder-decoders and LLMs through minor ablation studies.

4.1 Problem Definition

Suppose the feature space $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N\} \in \mathcal{X}$ is a set of *databases* where each instance $\mathbf{s}_i = \{\mathcal{T}_1, \ldots, \mathcal{T}_{m_i}\}$ is a collection of *representations*, each with its own set of *records* $\{\mathbf{r}_{j,1}, \ldots, \mathbf{r}_{j,n_j}\}$. Note that these representations can be graphs, tables, charts, etc. and that each database does not restrict itself to a specific type. Given the corresponding set of responses $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathcal{Y}$ where each response $\mathbf{y}_i = y_i^{(1)} \cdots y_i^{(k_i)}$ is a sequence of natural language tokens, the probabilistic model with parameters $\boldsymbol{\theta}$ is

$$\mathbb{P}\left[\mathbf{y}_i | \mathbf{s}_i; \boldsymbol{\theta}\right] = \prod_{m=1}^{k_i} \mathbb{P}\left[y_i^{(m)} | y_i^{(1)}, \ldots, y_i^{(m-1)}, \mathbf{s}_i; \boldsymbol{\theta}\right]$$

We informally define *large data-to-text generation* as the case where $\sum_{j=1}^{m_i} n_j$ is very large (in practice this will be in the thousands). Specifically, the input databases have a large number of records that need to be considered and "selected" as part of the model pipeline. As a result, the generated summaries $\hat{\mathbf{y}}_i$ will only discuss a small subset of the data records.

Specific to the ROTOWIRE dataset (and future variants introduced in this chapter), each record $\mathbf{r}_j$ can be represented as a tuple $\mathbf{r}_j = (e_j, v_j, u_j)$ representing the entity (i.e. player or team), value (i.e. numerical statistic, location, etc.), and relation (i.e. points, assists, date, etc.). In practice, an explicit relation u_j won't always exist so each record is either an entity-value pair or a relation object where the relationship defines an extra characteristic of the entity-value pair.

4.2 Datasets

We introduce four novel variants of the original ROTOWIRE dataset below. The first variant is a *cleaned* version of the original that accounts for trivial normalization and updates a considerable number of the summaries that were updated since its inception. The *second* introduces the supplementary data records as described earlier. The *third* variant further resamples and augments the previous to account for data contamination and newer examples between the last dated game and the most recent season end (2022 as of writing). The *fourth* variant is an extension of the previous which replaces the input records with a single image that embeds the data structures and layout-sensitive text formats (i.e. PDF page).

All of the examples were first curated and crawled from their appropriate web sources before processing in memory. For example, even with the original dataset we crawled

and extracted the summaries again from the ROTOWIRE source before normalizing and mapping the IDs back to their respective train-test partitions.

4.2.1 Rotowire (Cleaned)

This is the original version of the dataset as presented in Wiseman et al. (2017) with minor corrections to ensure that the results are comparable with the other experiments to be discussed. The data partitions and input features are preserved, and the changes are listed as follows.

- All the summaries are replaced with versions taken directly from the original web source at the time of writing. Most of the examples are not affected, however there were a select few with atypical writing styles that have since been updated to be more consistent with the others.
- Numbers in a lexical format are normalized into their digit values (i.e. ten is converted to 10)[1]. Note that this normalization didn't affect all of the tokens as we would have preferred however it was sufficient in practice.
- All dates are normalized to be listed in full instead of its abbreviated form. For example, `2015_12_25` or `Dec 12, 2015` are both standardized to `Friday, December 25, 2015`.
- The original summaries from Wiseman et al. (2017) were provided as a list of tokens instead of a raw string. To ensure that we can adopt existing LLMs during the experiments we opted to "detokenize" and revert to its original text through string heuristics. Most LLMs are based on a SentencePiece tokenizer which is sensitive to whitespace and punctuation placement thus it was imperative that the tokens were reverted to preserve formatting (preliminary experiments without this step resulted in significantly degraded performance of 9-10 BLEU points).

4.2.2 Rotowire (Supplemented)

To incorporate additional entities, the data table **s** needs to be replenished from another source[2] which is then mapped back to ROTOWIRE using team names and dates. The original input records are comprised of preliminary statistics and do not contain other relevant

[1] `https://github.com/allo-media/text2num`

[2] `http://site.api.espn.com/apis/site/v2/sports/basketball/nba/summary?event=` where the *event number* represents unique game identifiers.

information such as game venue, date, team totals (Wang, 2019), etc., let alone more sophisticated relations. We restrain the additional information to live-game statistics (i.e. those which are tracked during an NBA game), aggregated versions of the previous, and next-game information. For the sake of consistency, no data discussing in-game or historical events are included (i.e. such as play-by-play data, preview articles, etc.). After adding the supplementary records the final distribution yielded 82 possible input entities/relations as opposed to the original 27.

Admittedly, this is still not an exhaustive verification and input supplementation will remain a future area of research as it will also be aligned with work in efficient sequence processing for LLMs and alternate representations. For additional information on the specific entities refer to Section 4.3.3.

4.2.3 Rotowire (Full)

Another point worth noting is that the original data splits for ROTOWIRE are not disjoint. That is, around 25% of the datapoints in each of the valid and test partitions are also present in the training data. This could contribute to inflated metrics as the inclusion of additional data records in the input will encourage the highly parameterized PLMs to memorize the outputs conditioned on specific inputs. To counteract this the full dataset was resampled from their respective sources for the full seasons 2014-2022 to yield 10,169 examples. The data was partitioned into mutually exclusive train/validation/test sets of respective sizes 8135/1017/1017. The training data is approximately 2.4 times larger than the original dataset and can help determine if any issues seen with earlier work are alleviated by simply using a larger and more diverse training set. The input features are the same extended data records as discussed in the previous iteration (thus the records have to be sampled separately and mapped back to the summaries).

Note that this is presented as a separate dataset primarily for comparison sake. For example, ROTOWIRE (SUPPLEMENTED) preserves the original data splits to give a better idea of how supplementary data can improve/degrade performance independent of the model.

4.2.4 Rotowire (Vision)

Visual data-to-text generation is a specific case of DTG (or large DTG) where the data structures are embedded within images. This is closely related to other document understanding tasks such as *visual document understanding* where the models are required to

understand and interpret the contents within the document as part of its inference. Visual DTG however is not well-studied in the literature as a standalone task but has been implicitly incorporated in a few other large-scale reasoning benchmarks.

For each game in the ROTOWIRE (FULL) dataset, there is an associated game book created officially by the league which includes details about the game at various granularities[3]. For example, there are data tables for the full game, each intermediate period, etc. To construct the visual input for this dataset, we use the first page of each of these documents as it consists of all the relevant summary statistics as constructed in previous variants. The main difference is that the data itself is formatted into tables or aligned text (i.e. titles, headings, footnotes, etc.) within the input images.

There are some discrepancies between the data in the official game document and what was included in the earlier experiments. For example, details about upcoming games are added as aligned text in areas of extended white space. Refer to Figure 4.1 for a sample input with the additional features incorporated into the image. This specific dataset for conducting "visual" data-to-text generation will be challenging as there are various structured data formats to incorporate within a single image. Furthermore there are multiple font sizes (especially with the footnotes describing more nuanced instances) based on time period and page section which can make it difficult to finetune a model at lower resolutions. It will be worthwhile in future work to reconstruct these images in a standardized manner.

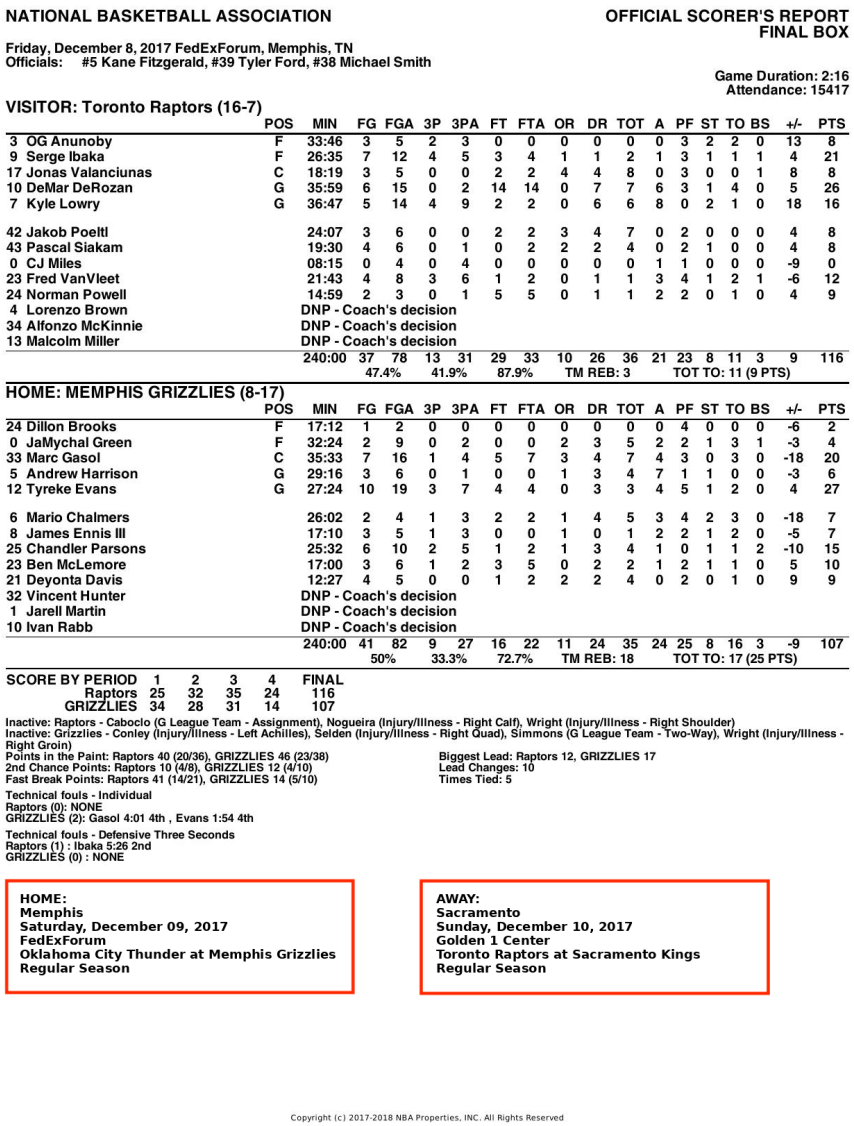
The motivation for experimenting with visual representations here and benchmarking its feasibility is twofold: (1) images preserve the implicit hierarchical and layout information for data structures such as tables (i.e. rows, columns, etc.) (Rebuffel et al., 2019); and (2) based on the dataset and resolution, the corresponding input sequence can be considerably shorter than if it were cast as a sequence of text tokens.

4.3 Evaluating Correctness

One of the primary obstacles in natural language generation (NLG) is verifying the accuracy of the model output. For example, classification and regression are paired with a plethora of automatic metrics that are simple to compute and offer a comprehensive overview of model performance (i.e. accuracy, precision, recall, etc.).

However, in the context of text generation one needs to evaluate both the language fluency and the domain correctness. It is imperative for the statistics, entities, and relationships described in the corresponding generated summary to be accurate. Furthermore,

[3] `https://www.nba.com/stats/gamebooks`



NATIONAL BASKETBALL ASSOCIATION

OFFICIAL SCORER'S REPORT
FINAL BOX

Friday, December 8, 2017 FedExForum, Memphis, TN
Officials: #5 Kane Fitzgerald, #39 Tyler Ford, #38 Michael Smith

Game Duration: 2:16
Attendance: 15417

VISITOR: Toronto Raptors (16-7)

	POS	MIN	FG	FGA	3P	3PA	FT	FTA	OR	DR	TOT	A	PF	ST	TO	BS	+/-	PTS
3 OG Anunoby	F	33:46	3	5	2	3	0	0	0	0	0	0	3	2	2	0	13	8
9 Serge Ibaka	F	26:35	7	12	4	5	3	4	1	1	2	1	3	1	1	1	4	21
17 Jonas Valanciunas	C	18:19	3	5	0	0	2	2	4	4	8	0	3	0	0	1	8	8
10 DeMar DeRozan	G	35:59	6	15	0	2	14	14	0	7	7	6	3	1	4	0	5	26
7 Kyle Lowry	G	36:47	5	14	4	9	2	2	0	6	6	8	0	2	1	0	18	16
42 Jakob Poeltl		24:07	3	6	0	0	2	2	3	4	7	0	2	0	0	0	4	8
43 Pascal Siakam		19:30	4	6	0	1	0	2	2	2	4	0	2	1	0	0	4	8
0 CJ Miles		08:15	0	4	0	4	0	0	0	0	0	1	1	0	0	0	-9	0
23 Fred VanVleet		21:43	4	8	3	6	1	2	0	1	1	3	4	1	2	1	-6	12
24 Norman Powell		14:59	2	3	0	1	5	5	0	1	1	2	2	0	1	0	4	9
4 Lorenzo Brown		DNP - Coach's decision																
34 Alfonzo McKinnie		DNP - Coach's decision																
13 Malcolm Miller		DNP - Coach's decision																
		240:00	37	78	13	31	29	33	10	26	36	21	23	8	11	3	9	116
			47.4%		41.9%		87.9%		TM REB: 3				TOT TO: 11 (9 PTS)					

HOME: MEMPHIS GRIZZLIES (8-17)

	POS	MIN	FG	FGA	3P	3PA	FT	FTA	OR	DR	TOT	A	PF	ST	TO	BS	+/-	PTS
24 Dillon Brooks	F	17:12	1	2	0	0	0	0	0	0	0	0	4	0	0	0	-6	2
0 JaMychal Green	F	32:24	2	9	0	2	0	0	2	3	5	2	2	1	3	1	-3	4
33 Marc Gasol	C	35:33	7	16	1	4	5	7	3	4	7	4	3	0	3	0	-18	20
5 Andrew Harrison	G	29:16	3	6	0	1	0	0	1	3	4	7	1	1	0	0	-3	6
12 Tyreke Evans	G	27:24	10	19	3	7	4	4	0	3	3	4	5	1	2	0	4	27
6 Mario Chalmers		26:02	2	4	1	3	2	2	1	4	5	3	4	2	3	0	-18	7
8 James Ennis III		17:10	3	5	1	3	0	0	1	0	1	2	2	1	2	0	-5	7
25 Chandler Parsons		25:32	6	10	2	5	1	2	1	3	4	1	0	1	1	2	-10	15
23 Ben McLemore		17:00	3	6	1	2	3	5	0	2	2	1	2	1	1	0	5	10
21 Deyonta Davis		12:27	4	5	0	0	1	2	2	2	4	0	2	0	1	0	9	9
32 Vincent Hunter		DNP - Coach's decision																
1 Jarell Martin		DNP - Coach's decision																
10 Ivan Rabb		DNP - Coach's decision																
		240:00	41	82	9	27	16	22	11	24	35	24	25	8	16	3	-9	107
			50%		33.3%		72.7%		TM REB: 18				TOT TO: 17 (25 PTS)					

SCORE BY PERIOD	1	2	3	4	FINAL
Raptors	25	32	35	24	116
GRIZZLIES	34	28	31	14	107

Inactive: Raptors - Caboclo (G League Team - Assignment), Nogueira (Injury/Illness - Right Calf), Wright (Injury/Illness - Right Shoulder)
Inactive: Grizzlies - Conley (Injury/Illness - Left Achilles), Selden (Injury/Illness - Right Quad), Simmons (G League Team - Two-Way), Wright (Injury/Illness - Right Groin)
Points in the Paint: Raptors 40 (20/36), GRIZZLIES 46 (23/38)
2nd Chance Points: Raptors 10 (4/8), GRIZZLIES 12 (4/10)
Fast Break Points: Raptors 41 (14/21), GRIZZLIES 14 (5/10)

Biggest Lead: Raptors 12, GRIZZLIES 17
Lead Changes: 10
Times Tied: 5

Technical fouls - Individual
Raptors (0): NONE
GRIZZLIES (2): Gasol 4:01 4th , Evans 1:54 4th

Technical fouls - Defensive Three Seconds
Raptors (1) : Ibaka 5:26 2nd
GRIZZLIES (0) : NONE

HOME:
Memphis
Saturday, December 09, 2017
FedExForum
Oklahoma City Thunder at Memphis Grizzlies
Regular Season

AWAY:
Sacramento
Sunday, December 10, 2017
Golden 1 Center
Toronto Raptors at Sacramento Kings
Regular Season

Figure 4.1: An example of an input image describing the summary statistics for a match in the ROTOWIRE dataset. Note the various data structures and whitespace formats embedded in a single input instance. The text included inside of the red boxes were added on top of the image itself. Note that the actual input doesn't include the red bounding box, it is merely for illustrative purposes.

any generated phrases not in the reference must also be factually grounded. This is difficult to optimize directly, however previous work demonstrated that it can be learned if the dataset is sufficiently large and diverse. Using a LLM also alleviates the need to learn language intricacies and instead focus on content selection which encompasses this problem of truthfulness.

4.3.1 Relation Extraction for Evaluation

Wiseman et al. (2017) and Puduppully et al. (2019*b*) devised an automated evaluation system using information extraction models similar to (Collobert et al., 2011; Zeng et al., 2014; dos Santos et al., 2015; Zhou et al., 2008; Zhang, 2004). Specifically, given a collection of all possible entities e and numeric values m in each sentence, the extraction system models $\mathbb{P}[r.t|e, m; \boldsymbol{\theta}]$ for every pair (if no relationship exists, $r.t = \epsilon$).

The final models were trained on the original game summaries with algorithmic label construction. The entity-value pairs are collected from each sentence and mapped back to the database to determine any possible relationship. They omit any pair that can be mapped to multiple relations. As the dataset will have noisy labels, the models will ideally have high precision with lower recall. The original IE models reported an accuracy and recall of 90% and 60% respectively. They were later updated in Puduppully et al. (2019*a*) to include textualized numbers with resulting accuracy and recall of 94% and 80% respectively.

The original extraction models however were limited to a small subset of easily verifiable numeric boxscore records (i.e. team points, player points, player assists, etc.) shown in Figure 4.2. There are additional entities to verify such as dates, locations, winner/loser, game streaks, etc. Note that in order to extend these models to account for the additional entities, the actual record collection r needs to be revised to include such relationships (otherwise the extractions cannot be mapped back to verify correctness).

To get a better idea of the various relationships that can be evaluated, we reviewed thirty randomly sampled summaries from ROTOWIRE (FULL) and annotated specific relations and entities that can be mapped back to the supplemented in records (listed in Figure 4.3). This yields a new total of 82 unique entity relations to verify and includes trivial ones such as locations, venues, accumulated values, etc.

It was also found that some values can be implicit representations of the same statistic. For example, rather than expressing that Team A scored p points in the quarter, the summary may state that Team A leads/trails Team B by q points (either the accumulated

Player Types	PLAYER-PTS	PLAYER-REB	PLAYER-AST	PLAYER-STL	PLAYER-BLK	PLAYER-MIN
	PLAYER-FGA	PLAYER-FGM	PLAYER-FTM	PLAYER-FTA	PLAYER-FG3M	PLAYER-FG3A
	PLAYER-TO	PLAYER-OREB	PLAYER-PF	PLAYER-FG_PCT		
Team Types	TEAM-AST	TEAM-WINS	TEAM-LOSSES	TEAM-PTS	TEAM-REB	TEAM-FG_PCT
	TEAM-FG3_PCT	TEAM-PTS_QTR1	TEAM-TOV	TEAM-PTS_QTR4	TEAM-FT_PCT	

Figure 4.2: The original 27 entity relations found in the extraction dataset of Puduppully et al. (2019*a*).

or per-quarter score). This specific example is not adjusted in the inputs as it gives a way to gauge if the LLMs are able to quantitatively reason within these datasets.

In the next section, we propose two improvements to the relation extraction approach described above. The first is to reformulate extraction as a "text-to-text" problem such that we can incorporate generative LLMs. The second is to further supplement the extraction models to include additional entities and relationships as discussed.

4.3.2 Extracting Relations with LLMs

The key idea to using LLMs for extraction tasks is to treat them in a "text-to-text" fashion (Raffel et al., 2019). The specific formulation, inspired by Athiwaratkun et al. (2020), is to map the input back to a modified version of itself where the entities and relations are wrapped (or "tagged") around the substring(s) of interest. This technique provides the added benefit of recovering the locations of where an entity or relation is flagged at the cost of longer output sequences. Refer to Figure 4.4 for an example on how the input and output sequences are formulated.

Another benefit specific to the "text-to-text" method is that it provides a means to process the entities/relations as part of the model inference. For example, the phrase containing `...scored one point...` can be mapped to `...scored [1 | PT] point...` which automatically converts textual representations of numbers (and thus taking advantaging of inheritted knowledge within LLMs). A more complicated example is to standardize `...won four straight...` to `...won [W4 | GAME-STREAK] straight...`. This alleviates the need for a dedicated postprocessing function which can be tedious to design. Furthermore, in this specific case the process of verifying entity precision is simplified to

Player Types	PLAYER-POS	STARTER	PLAYER-FGM	PLAYER-FGA
	PLAYER-FG_PCT	PLAYER-3PTM	PLAYER-3PTA	PLAYER-3PT_PCT
	PLAYER-FTM	PLAYER-FTA	PLAYER-FT_PCT	PLAYER-MIN
	PLAYER-OREB	PLAYER-DREB	PLAYER-REB	PLAYER-AST
	PLAYER-STL	PLAYER-BLK	PLAYER-TO	PLAYER-PF
	PLAYER-PLUS_MIN	PLAYER-PTS	BENCH	
Team Types	REG-WINS	REG-LOSS	TEAM-PTS	TEAM-Q1_PTS
	TEAM-Q2_PTS	TEAM-Q3_PTS	TEAM-Q4_PTS	TEAM-Q1_PTS_TOT
	TEAM-Q2_PTS_TOT	TEAM-Q3_PTS_TOT	TEAM-WINS	TEAM-LOSS
	HOME-WINS	HOME-LOSS	ROAD-WINS	ROAD-LOSS
	NEXT-DAY	NEXT-DATE	NEXT-VENUE	NEXT-CITY
	NEXT-TEAM	TEAM-FGM	TEAM-FGA	TEAM-FG_PCT
	TEAM-3PM	TEAM-3PA	TEAM-3P_PCT	TEAM-FTM
	TEAM-FTA	TEAM-FT_PCT	TEAM-TREB	TEAM-OREB
	TEAM-DREB	TEAM-AST	TEAM-STL	TEAM-BLK
	TEAM-TO	TEAM-TTO	TEAM-TOT_TO	TEAM-TECH
	TEAM-TOT_TECH	TEAM-FLAG	TEAM-TO_PTS	TEAM-FB_PTS
	TEAM-PIP	TEAM-FLS	TEAM-LL	TEAM-OT1_PTS
	TEAM-Q4_PTS_TOT	TEAM-OT2_PTS	TEAM-OT3_PTS	TEAM-OT1_PTS_TOT
	TEAM-OT2_PTS_TOT	TEAM-OT4_PTS	TEAM-OT3_PTS_TOT	
Game Info	VENUE	GAME-CITY	GAME-DAY	FULL-DATE

Figure 4.3: All the possible data-grounded entities found in a subset of summaries as described in Section 4.3.1. Italicized entities are part of the original collection of entities captured in the Rotowire automatic evaluation models. The starred entities are those we found in the summaries that can be incorporated into updated evaluation models.

simply collecting the wrapped substring(s). However, there is still a need to post-process factitious or invalid outputs due to the *hallucinogenic* side-affect of generative decoders.

Example 1.

Input.
The Toronto Raptors defeated the visiting Golden State Warriors, 118-109, in Game 1 of the NBA Finals at Scotiabank Arena on Thursday.

Output.
The [Toronto Raptors | TEAM] defeated the visiting [Golden State Warriors | TEAM], [118 | TEAM-PTS | Toronto Raptors]-[109 | TEAM-PTS | Golden State Warriors], in Game 1 of the NBA Finals at Scotiabank Arena on Thursday.

Example 2.

Input.
The Raptors (1-0) were led by Pascal Siakam, as he collected 32 points, eight rebounds, five assists and two blocks on 14-of-17 from the field.

Output.
The [Raptors | TEAM] ([1 | TEAM-WINS | Raptors]-[0 | TEAM-LOSSES | Raptors]) were led by [Pascal Siakam | PLAYER], as he collected [32 | PLAYER-PTS | Pascal Siakam] points, [eight | PLAYER-REB | Pascal Siakam] rebounds, ...

Figure 4.4: Two examples of how the input and output texts will look under the two "text-to-text" formulations described in Section 4.3.2 using the original auto-evaluation entities from Figure 4.2. The first approach has the benefit of shorter output sequences and will be more efficient to train. The second approach allows for mapping the location of each extraction. Note that the entities `TEAM` and `PLAYER` are added during training to help the model learn relationships. These are omitted during post-processing.

4.3.3 Data Setup and Supplementary Relations

The data curation and update procedure is large described in Section 4.2. This is the same collection of relations that will be mapped back to the databases during the evaluation process. The full distribution of all 82 entities is given in Figure 4.3.

To create the dataset, we consider the ROTOWIRE (FULL) dataset and tokenize the sentences (Bird, 2004). The additional data was necessary to extend the empirical distributions for augmentation (discussed in the next section). We also added additional heuristics

such that sentences without personified entities (i.e. they only appear as pronouns) are combined with the prior sentence. Furthermore, we normalized the numbers that appear in textual form into their numerical equivalent (i.e. `ten` becomes `10`).

4.3.4 Data Augmentation

The inclusion of additional entities introduces an increased amount of noise when automating the label assignment. One way to counteract this is to use key-phrase heuristics to filter out ambiguous relationships. However, this still doesn't guarantee a sufficient amount of the data will be labelled correctly. To ensure that the relation-based metrics have maximum accuracy, a small subset of the sentences are manually annotated first and an augmentation scheme is employed to inflate the training set.

The augmentation pipeline is to randomly replace key entities (i.e. players, teams, cities, venues, etc.) and values in the sentence while preserving the annotated relationship. This approach is simple, however has proven effective due to the consistent language and phrasing in the summaries. The dedicated held-out set is only from annotated or human-verified data whereas augmentation is applied to the training and validation sets. The replaced entities and values were generally kept within the features' empirical distribution.

4.3.5 Metrics for Evaluating Extractions

Three common metrics used to evaluate any sort of "information retrieval" system are the *precision*, *recall*, and *F1* measures. Within the context of information extraction, consider the model f that takes an input document $\mathbf{x}$ and returns $\widehat{\mathcal{E}}$, the set of extracted entities. If $\mathcal{E}$ is the corresponding set of true entities (i.e. the set of entities that actually correspond to $\mathbf{x}$), then

$$\text{precision} = P = \frac{|\widehat{\mathcal{E}} \cap \mathcal{E}|}{|\widehat{\mathcal{E}}|}, \qquad \text{recall} = R = \frac{|\widehat{\mathcal{E}} \cap \mathcal{E}|}{|\mathcal{E}|}, \qquad \text{F1} = F_1 = 2\left(\frac{P \cdot R}{P + R}\right)$$

Note that the F1 measure is just the harmonic mean of both the precision and recall.

Within the context of the text-to-text relation extraction models from Section 4.3.2, consider the associated dataset $\mathcal{D} = \{(\mathbf{x}_1, \mathcal{E}_1), \ldots, (\mathbf{x}_n, \mathcal{E}_n)\}$. The network $f_a : \mathcal{V} \to \mathcal{V}$ takes an input $\mathbf{x}_i = x_{i,1} \cdots x_{i,m_1}$ of natural language tokens (from the vocabulary $\mathcal{V}$) and maps it back to another sequence of natural language tokens $\widehat{\mathbf{y}}_i = \widehat{y}_{i,1} \cdots \widehat{y}_{i,m_2}$ (like Figure 4.4). To construct $\widehat{\mathcal{E}}_i$, we use string heuristics to collect all relevant substrings, such as those

wrapped with `[...]`. The final step is to aggregate the sets before applying the measures in (4.3.5). That is, $\mathcal{E} = \cup_{i=1}^{n} \mathcal{E}_i$ and $\widehat{\mathcal{E}} = \cup_{i=1}^{n} \widehat{\mathcal{E}_i}$ in (4.3.5).

4.3.6 Experimental Results

We opted to use encoder-decoders for these models as the fine-tuning procedure was more aligned with pure extraction. The full updated auto-evaluation results on the held-out set (approximately 500 observations) are provided in Table 4.1 for various generative models. The models were optimized using Adafactor with relative step sizes for a maximum of five epochs. The best model checkpoint was selected using beam search with $k = 5$ over the validation set.

	Greedy Decoding			Beam Search ($k = 5$)		
Model	*Precision*	*Recall*	*F1*	*Precision*	*Recall*	*F1*
T5-small	93.20	92.62	92.91	93.39	92.98	93.18
T5-base	94.22	94.13	94.17	94.44	94.17	94.30
T5-large	95.32	93.77	94.54	**95.61**	93.93	**94.76**
BART-base	36.98	36.07	36.52	39.68	32.46	35.71
BART-large	35.86	34.05	34.93	55.35	32.94	41.30
Pegasus	92.80	94.17	93.48	92.98	**94.56**	93.76

Table 4.1: Table of metrics evaluating the efficacy of the proposed generative extraction models on the updated autoeval dataset with 82 total entities. The best metric is indicated in boldface. Details on calculating the precision, recall, and F1 measures are described in Section 4.3.5.

Empirically, the "text-to-text" approach is feasible based on how well most of the generative models performed (apart from BART). For those that were able to generalize, it appears that they excelled in the extraction task regardless of their size and decoding strategy. That is, increasing the size and number of beams only marginally improves performance. This allows one to use a smaller model with simpler decoding in practice to quickly curate the extractions as opposed to relying on the larger variants for higher precision.

4.3.7 Evaluating Text Generations

For the next couple of sections the base T5 autoeval model with greedy decoding is used to evaluate the statistical correctness of the generated texts. Note that the evaluation

approach described below is similar to what was discussed in Section 3.2.3.

To proceed with evaluation, let $y_{1:T_1}, \hat{y}_{1:T_2}$ be the two documents of interest (in this case, the gold and predicted summaries respectively). Using the auto-evaluation extraction model $f_a(\mathbf{x})$ as defined in Section 4.3.5, we first collect the *ordered* set of extractions from each document. Suppose $\mathcal{E} = \{e_1, \ldots, e_{m_1}\}$ and $\widehat{\mathcal{E}} = \{\widehat{e}_1, \ldots, \widehat{e}_{m_2}\}$ denote these sets accordingly. To gauge the correctness of the generated text, we calculate the following:

- *Content Selection (CS):* compute the *precision* (P) and *recall* (R) between the unique relations extracted from $\hat{y}_{1:T_1}$ and $y_{1:T_2}$. That is, we evaluate

$$P = \frac{|\mathcal{E} \cap \widehat{\mathcal{E}}|}{|\widehat{\mathcal{E}}|}, \qquad R = \frac{|\mathcal{E} \cap \widehat{\mathcal{E}}|}{|\mathcal{E}|}$$

  This is intended to measure whether the content of the generated summary matches what is present in the gold counterpart.

- *Relation Generation (RG):* compute the *precision* (P) and *number* of unique relations (#) extracted from $\hat{y}_{1:T_1}$ that appears in the database $\mathbf{s}$. If $\mathbf{s}$ is the set of all possible entities or relations associated to the game, then

$$\# = |\widehat{\mathcal{E}}|, \qquad P = \frac{|\mathbf{s} \cap \widehat{\mathcal{E}}|}{|\widehat{\mathcal{E}}|}$$

  The number of relations gives a measure for the amount of content present in the summaries. Similarly, the precision determines whether all the generated content is factually grounded, particularly those that do not appear in the gold summaries.

- *Content Ordering (CO):* compute the normalized Damerau-Levenshtein distance (DLD) (Brill and Moore, 2000) between the *sequence of records* extracted from $\hat{y}_{1:T_1}$ and $y_{1:T_2}$. That is,

$$\text{DLD\%} = \text{Damerau-Levenshtein}(\mathcal{E}, \widehat{\mathcal{E}})$$

  As stated earlier, we require that $\widehat{\mathcal{E}}$ and $\mathcal{E}$ are ordered sets. The DLD metric takes two strings and essentially determines the number of operations needed to transform one into the other. This provides a measure for how similar the content ordering is between the generation and gold summaries. We use DLD here to be consistent with Wiseman et al. (2017).

Wiseman et al. (2017) further conducted studies involving human evaluators to determine if the quantities above were consistent with the opinion of raters. For the most part, they found that the precision metrics correlated well with humans. However, similar tests will need to be done now as the models and entities are largely revised.

4.4 Experimental Setup

This section applies different encoder-decoders on the datasets described in Section 4.2 to establish appropriate baseline scores. The ROTOWIRE (CLEANED) and ROTOWIRE (SUPPLEMENTED) are used first to compare the effect of supplementary input records whereas the ROTOWIRE (FULL) and ROTOWIRE (VISION) datasets are used to evaluate the differences and limitations of the vision and text inputs. The specific pre-processing steps taken before applying the models are discussed in the next section. For all experiments, we follow Ethan Joseph and Si (2021) and report the supplemented extraction metrics along with BLEU, METEOR, and ROUGE. We leverage the n-gram scores over the summaries to provide a similarity measure for the surrounding language between the two on top of the actual entities and relations.

4.4.1 Data Processing and Experimental Setup

For each game, the associated box score data must first be converted into a text sequence that will be fed as input to an LLM. This process of casting a multi-dimensional collection of records into a sequence is known as *linearization*. Unlike the strategy incorporated by Kale (2020), each of the ROTOWIRE variants have a large number of unique records that requires strategically converting the inputs into an intermediate medium first.

Following Kale (2020), Ethan Joseph and Si (2021) created new tokens and appended them to the model vocabulary to act as separators between different entities. This is an intuitive approach however it will considerably increase the vocabulary space especially if there are a large number of unique entities. Another option is to use natural language separators (for example in English, the semicolon, colon, pipe, etc.) which are already present in the vocabulary with initialized weights. To tokenize the data records, we use a combination of English separators to form logical groups that are separated with the pipe character, "—" (i.e. first tokenize game information in one group, then team statistics, then player level statistics). Figure 4.5 provides an example of tokenizing and combining the separate groups.

As there are still a significant number of input tokens, we will work almost exclusively with LLM encoder-decoders implementing ideas from Section 3.1.1. The exception is T5-SMALL which is small enough where quadratic complexity doesn't exhaust our available compute memory.

Processing ROTOWIRE (VISION) is more straight-forward and is largely confined to rescaling the inputs to different resolutions. The first resolution 384×384 is a common

Original Input:

```
{
   “home_team” : “Toronto Raptors”,
   “away_team” : “Milwaukee Bucks”,
   “home_line” : [18, 25, 28, 19],
   “away_line” : [31, 19, 26, 18],
   “home_pts” : 100,
   “away_pts” : 94,
   …
}
```

Tokenized Input:

```
|HOME;Toronto Raptors;100;18,25,28,19
|AWAY;Milwaukee Bucks;94;31,19,26,18
```

Figure 4.5: A simplified example of how the raw data records in an arbitrary object are tokenized and converted into a single sequence of text tokens. Note that natural language separators are used to “separate” different groups of statistics contingent on domain knowledge. In this case, the separation is applied to the different team and score types.

input size for many vision-based transformers (though not for granular CV tasks such as segmentation). The other aspect ratios are $\frac{1}{2}$ and $\frac{2}{3}$ the original input images'. A special prompt token `<s>` was assigned as the initial decoder input to match the prompt style fine-tuning of Kim et al. (2022). Specifically, the gold summaries were wrapped with special `<s>`, `</s>` tokens signifying the start and end of the generated summary.

All of the models were trained with Adafactor and batch size of 4 on a single A6000 GPU. The best checkpoint was selected using the validation set and all outputs were generated using beam search with $k = 5$. For decoding in general, the maximum length was restricted to the 95th quantile of the gold summaries' token length distribution derived from the training set and we do not incorporate any additional loss or n-gram penalties. Furthermore we do not leverage any copy or equivalent correction mechanisms to fix values/statistics. All the methods proposed are purely derived through end-to-end inference

(which differ largely from all modelling work in Section 3.2.3).

4.4.2 Results

The results for all four datasets are presented in Table 4.2. The first three were fit using LLMs that adapted sparse attention mechanisms whereas the last vision dataset was fit using DONUT (discussed in Section 3.1.5) at differing resolutions. Furthermore we fit the vision dataset with and without the supplementary entities to gauge any immediate effects of doing so.

The remaining sections of this chapter analyze different relationships between the datasets (i.e. effect of including supplementary features, vision v.s. text feasibility, etc.). For presented generations, specific phrases are recoloured based on factual correctness. Green and red text indicates factually correct and incorrect statements that can be mapped to the input data records respectively. Purple and blue text represent factually correct and incorrect statements that are *not* accompanied with data records in the input respectively.

4.4.3 Anaylsis: Sparse Attention and Updated Autoeval

The performance increase between the original and supplemented datasets are more substantial than the reported average 1-4 points per statistic from Wang (2019). In the first three datasets, the sparse variants of the encoder-decoders are largely competitive or outperform the full-attention T5-SMALL especially with more training data. Furthermore, most of the models were specifically designed to generalize over downstream summarization but easily adjusted their encoder to the tokenized sequences of Section 4.2. As an aside, BIGBIRD performed the worst relatively but this is likely due to the checkpoint used (i.e. it was further finetuned on a specific summarization dataset (Cohan et al., 2018)).

The impact of the replenished autoeval models is particularly clear across the first two datasets. However, in the vision dataset note the increased precision and recall between the fits when including information that wasn't originally present in the boxscore. A significant proportion of the missing entities were related to *next game scheduling* which only appears as a short sentence or two. As expected, the n-gram metrics are largely unchanged providing a false sense of adequacy. If the extra features were not incorporated into the autoeval encoder-decoder then the other metrics would also exhibit similar misleading behaviour.

Rotowire (Clean)								
Model	**RG**		**CS**		**CO**	**BLEU**	**ROU-L**	**METR**
	#	**P**	**P**	**R**	**DLD%**			
T5-small	42.21	**88.64**	34.83	35.77	19.05	19.94	27.94	32.75
T5-base	47.43	88.41	34.43	**39.72**	20.54	21.68	29.25	34.83
LongT5-base	43.32	88.51	**36.52**	38.49	**22.43**	**23.12**	**30.96**	**35.40**
PEGASUS	**52.18**	75.63	24.61	31.24	13.49	17.44	25.62	30.62
LED-base	47.30	85.60	32.05	36.87	18.66	20.85	27.85	33.69
Rotowire (Supplemented)								
T5-small	46.24	**93.78**	36.83	41.43	22.03	21.74	29.56	35.03
LongT5-base	41.08	92.57	**46.71**	**46.68**	**32.99**	**32.74**	**39.87**	**43.36**
BigBird	43.71	52.48	24.88	26.46	19.51	27.86	34.36	40.08
LED-base	**47.49**	89.37	35.69	41.23	23.02	25.25	31.31	38.13
Rotowire (Full)								
T5-small	47.53	95.45	**47.50**	**53.08**	**28.39**	29.32	**36.90**	41.98
LongT5-base	**48.23**	**95.74**	46.81	**53.08**	27.26	**29.41**	36.47	**42.23**
BigBird	46.87	82.09	39.98	44.05	21.82	27.21	34.26	40.66
LED-base	47.42	92.80	44.44	49.54	26.12	28.82	36.11	41.49

Rotowire (Vision)								
Res.	**RG**		**CS**		**CO**	**BLEU**	**ROU-L**	**METR**
	#	**P**	**P**	**R**	**DLD%**			
Original Input Image								
384 × 384	15.85	13.51	1.49	0.55	0.41	9.12	24.19	27.77
826 × 638	44.62	84.84	39.48	41.42	19.30	25.22	33.44	38.27
1100 × 850	**47.21**	88.78	42.61	47.29	21.94	26.96	34.22	39.45
Incl. Supplementary Features								
826 × 638	45.83	88.64	40.76	43.91	20.97	25.98	33.53	39.39
1100 × 850	44.51	**94.09**	**46.56**	**48.73**	**24.32**	**26.98**	**34.90**	**39.75**

Table 4.2: Table of metrics evaluating the efficacy of the proposed generative extraction models and data processing pipelines on all four variants of the ROTOWIRE dataset as discussed in Section 4.2 under the proposed 82 entities autoeval scheme. The best metric for each dataset is identified with boldface. Note that *ROU-L* is shorthand for the ROUGE-L score and *METR* is shorthand for the METEOR score. All of the preceding metrics for evaluating entity relations are defined in Section 4.3.7.

4.4.4 Analysis: Impact of Supplementary Input Records

Figure 4.6 provides annotated examples of LongT5's generations from both test sets. Note that due to randomness the outputs will be similar but not identical.

The models from the original data are more prone to generation inaccuracies even for statistics or items directly present in the input. This could be due to lower confidence when selecting values to "copy" because of the excess missing statistics. In the third sentence, the model incorrectly accumulates the halftime scores however this is a widely reported issue with LLMs in general (Lewkowycz et al., 2022). Within domain context the model also fails to reason for example when identifying players who had "pair efforts" and was even unable to place enough attention in preventing repetitive statements (i.e. `Shaun Livingston`'s statistics are reported three times one after another). Though, the generations correctly identified the *conferences* and *divisions* of both teams which are not included in the input records. This demonstrates the LLMs ability to quickly "memorize" key phrases even during task-specific fine-tuning.

The supplemented models on the other hand are more stable and generate statistically grounded summaries. Consequently the generations display a better grasp of domain-specific reasoning such as recognizing key contributors. It also correctly mentioned how the Cavaliers will play the "second half of a back-to-back" as it recognized the current and next games are in following days. It is still not perfect and the generated text is prone to contradicting itself (i.e. claiming Golden State is struggling despite having a single loss).

4.4.5 Analysis: Impact of Larger Training Set

The results are no longer comparable due to different evaluation sets. Overall, all of the models benefited from additional training data and unbiased evaluation partitions. The *content selection* metrics are high (on average, around half of the relations that the model generates are shared with the gold summary) as well as the generated relations' precision. Furthermore the n-gram metrics are all very similar regardless of the architecture and size.

The average length of the generated summaries is considerably shorter and the content is more consistent. For example, the summary starts with the final game score and winner before describing the point distribution of either the team or key players after each period. It then lists out notable performances from both teams before concluding with their next opponents. This is also evident in the two generated examples from LongT5 presented in Figures 4.7.

The Golden State Warriors (28-1) defeated the Cleveland Cavaliers (19-8) 89-73 on Friday. The Cavaliers looked like 2 entirely different teams during the first and second halves while visiting Golden State. Down 58-47 at halftime and surrendering 37 second quarter points, the defensive breakdowns and sloppy fouls began to bury the Warriors early on. The Golden State defense also stymied Cleveland early, limiting them to 32 percent shooting and a paltry 17 percent success rate from beyond the arc. Kevin Love posted a second consecutive strong performance, finishing with 10 points (5-16 FG, 0-5 3Pt) and 18 rebounds over 39 minutes. Irving contributed 13 points (4-15 FG, 0-6 3Pt, 5-5 FT) and 3 rebounds. Tristan Thompson paced the Cleveland bench with 8 points (3-6 FG, 2-2 FT) and 10 rebounds over 27 minutes. The Cavaliers fell to 19-8 and remain atop the Eastern Conference's Central Division. They head to Los Angeles to face the Lakers on Sunday. Golden State was led by a pair of 22-point efforts from Draymond Green and Stephen Curry. Green also added 15 rebounds and 7 assists, while Curry pitched in with 7 rebounds and 7 assists. Shaun Livingston and Klay Thompson contributed a pair of 16-point efforts from the bench. Shaun Livingston was able to provide 16 points (8-9 FG) and 3 rebounds, while Shaun Livingston added 16 points (8-9 FG) and 3 rebounds. Golden State improved to 28-1 and remains atop the Western Conference's Pacific Division. They head to Los Angeles to face the Lakers on Sunday.

The Golden State Warriors (28-1) defeated the Cleveland Cavaliers (19-8) 89-83 on Friday.It's not too often that Golden State gets tested at home, but they gave it to the Cavaliers here. They just couldn't quite get it going tonight, but they were carried by a monster performance from Draymond Green. The former MVP tallied 22 points, 15 rebounds and 7 assists in a monster game and made an impact on loose balls and rebounds as well. His defense was the key to the win though, as the Cavaliers were held to just 32 percent shooting from the field and a meager 17 percent from three-point range. The vaunted defense wasn't shabby either, as they held the Cavaliers to 32 percent shooting and drew just 11 turnovers in the victory.Cleveland played arguably their best game of the season on Friday, but couldn't quite pull off a big win. LeBron James and Kevin Love were both monster in the loss, but no other player had more than 13 points. James finished with 25 points, 9 rebounds and 2 assists while Love recorded a double-double, totaling 10 points and a game-high 18 rebounds in the defeat. The only other player in double figures was Matthew Dellavedova, who scored 10 points off the bench. The Cavaliers will have to play again on Saturday, as they take on the Portland Trail Blazers in the second half of a back-to-back.It's tough for Golden State to win any game right now, but they'll have to figure something out over their next game, when they take on the Sacramento Kings on Christmas.

Figure 4.6: An example of two generations of the same match in the test set between the Rotowire (Clean) (top) and Rotowire (Supplemented) (bottom).

The Philadelphia 76ers defeated the host Atlanta Hawks, 104-99, in Game 6 of the Eastern Conference Semi Finals at State Farm Arena on Friday evening. The contest got out to a competitive start, as the Sixers held a two-point lead after one quarter. The Sixers kept it close through the second, but the hosts stayed on top and took a two-point lead into halftime. The Sixers carried the momentum into the third quarter, where they outscored the hosts, 33-25, to take a nine-point lead into the final frame. The Sixers stayed in control through the 4th and held on for the road win. Tobias Harris led the 76ers with 24 points, 5 rebounds and 2 assists, in 39 minutes. Seth Curry followed up with 24 points, 3 rebounds, 2 assists, a steal and a block, while Joel Embiid totaled 22 points, 13 rebounds and 2 blocks. The 76ers shot 41 percent from the field, including 12-of-29 from long range. Meanwhile, Trae Young led the Hawks with 34 points, 12 assists, 5 rebounds and 3 steals, in 39 minutes. Kevin Huerter added 17 points, 11 rebounds, 4 assists, a steal and a block, while Danilo Gallinari chipped in 16 points off the bench. The Hawks shot 41 percent from the field, including 10-of-31 from deep.

The Toronto Raptors defeated the host Memphis Grizzlies, 116-107, at FedEx Forum on Friday evening. The Grizzlies got out to a good start, leading 34-25 after one quarter. The Raptors responded in the second quarter, where they outscored the Grizzlies 32-28, to take a 59-50 lead into halftime. The Raptors carried the momentum into the third quarter, where they outscored the Grizzlies 35-31, to take a 86-75 lead into the final frame. The Raptors stayed in control through the 4th and cruised to their second straight win. DeMar DeRozan led the Raptors with 26 points, 7 rebounds, 6 assists and one steal, in 35 minutes. Serge Ibaka followed up with 21 points on 7-of-12 shooting, while Kyle Lowry tallied 16 points, 8 assists, 6 rebounds and 2 steals. The Raptors shot 48 percent from the field, including 13-of-31 from long range. Meanwhile, Tyreke Evans led the Grizzlies with 27 points on 10-of-19 shooting, in 27 minutes. Marc Gasol followed up with 20 points, 7 rebounds and 4 assists, while Chandler Parsons chipped in 15 points off the bench. The Grizzlies shot 46 percent from the field, including 9-of-27 from long range. The Grizzlies (8-17) will look to bounce back as they play host to the Thunder on Saturday. The Raptors (16-7) take on the Kings in Sacramento on Sunday.

Figure 4.7: Two generated examples from the fully replenished ROTOWIRE (FULL) dataset.

These models benefited from additional training examples by generating coherent phrases laced with correct domain reasoning. For example, the first summary correctly identifies that the game was competitive due to the close score splits after each period. Furthermore, the summaries correctly identify the "host" and "visiting" teams. Another instance

of "absorbed knowledge" is the correct usage of the 76ers' alternate name "Sixers." This is more apparent in large part because of the additional summaries using these symbols.

4.4.6 Analysis: Feasibility of Image Representations

For the smallest resolution 384×384, the model is unable to comprehend the data in the input and produces unrelated sequences. The model performs significantly better as the input size increases. This is expected due to the discrepancy on fonts and layout over the dataset. It is likely that the intermediate size was still too noisy to decipher *footer* font text about general team statistics.

Though the results are omitted here, fine-tuning a vision encoder-decoder directly from two pre-trained checkpoints (Li et al., 2021) without pre-training also failed to learn. This was consistent regardless of the image resolution and individual encoder/decoder strength. This indicates the need for a preliminary training objective as the current dataset size is insufficient. DONUT was trained over text recognition objectives allowing it to at least comprehend and incorporate document content.

Figure 4.8 provides some generated examples from the highest resolution model fit with supplementary data (note the same games as Figure 4.7).

The generations are able to capture embedded statistics and correctly substitute them into the output sequences. One notable issue is when discussing player minutes as the input displays them in the `<MM>:<SS>` format rather than rounded. The model ultimately learned to extract just the minutes due to examples with rounded down times and incorrectly learns it as a global pattern. This is applies to percentages.

Compared to the LLMs with the same amount of data, DONUT struggles with domain-specific reasoning and even basic understanding of its conditional generation. In the first example, the model is unable to identify that the game wasn't high-scoring and "leading" players. In the second example, the model correctly identified the Raptors defeated the Grizzles, however it incorrectly swapped the context for both teams for the remainder of the generation. This demonstrates inferior language understanding of VDU models without incorporating any reasoning objectives into pre-training. Here, DONUT excelled in content selection and relation generation but not in coherent and trivial generation.

The Philadelphia 76ers defeated the host Atlanta Hawks, 104-99, at State Farm Arena on Friday evening. The contest got out to a high-scoring start as the Hawks led, 29-22, after one quarter. The Sixers picked up the pace in the second and outscored the hosts, 25-22, in the quarter, to cut it to a three-point game by halftime. The Hawks came out stronger in the third and outscored the visitors, 33-25, in the quarter, to take a nine-point lead into the final frame. The Hawks kept up the fight in the 4th and outscored the visitors by 10 points in the quarter, to earn their second straight win. Joel Embiid led the 76ers with 22 points, 13 rebounds and 2 blocks, in 39 minutes. Tobias Harris added 24 points, 5 rebounds and 2 assists, while Tyrese Maxey chipped in 16 points, off the bench. The 76ers shot 40 percent from the field, including 12-of-29 from long range. Meanwhile, Trae Young led the Hawks with 34 points, 12 assists, 5 rebounds and 3 steals, in 40 minutes. Kevin Huerter added 17 points, 11 rebounds, 4 assists, a steal and a block, while Danilo Gallinari chipped in 16 points on 5-of-10 shooting. The Hawks shot 41 percent from the field, including 10-of-31 from long range. The Hawks (3-3) will look to bounce back as they take on the 76ers in Philly on Sunday. The 76ers (3-3) play host to the Hawks on Sunday.

The Toronto Raptors defeated the host Memphis Grizzlies, 116-107, at FedEx Forum on Friday evening. The Raptors got out to a good start on the road, with a, 34-25, lead after one quarter. The Grizzlies responded in the second quarter, where they outscored the Raptors, 32-28, to take a one-point lead into halftime. They carried the momentum into the third quarter, where they outscored the Raptors, 35-31, to take a nine-point lead into the final frame. The Raptors came out strong in the 4th and managed to tie it up at 110, with 3 minutes left in the contest. However, the Raptors came up clutch down the stretch to earn their second straight win. DeMar DeRozan led the Raptors with 26 points, 7 rebounds, 6 assists and one steal, in 35 minutes. Serge Ibaka followed up with 21 points, 2 rebounds, one assist, one steal and one block, while Kyle Lowry tallied 16 points, 8 assists, 6 rebounds and 2 steals. The Raptors shot 47 percent from the field, including 13-of-31 from long range. Meanwhile, Tyreke Evans led the Grizzlies with 27 points, 3 rebounds, 4 assists and one steal, in 27 minutes. Marc Gasol followed up with 20 points, 7 rebounds and 4 assists, while Chandler Parsons chipped in 15 points, 4 rebounds, 2 blocks and one assist. The Grizzlies shot 50 percent from the field, including 9-of-27 from long range. The Grizzlies (8-17) will look to bounce back as they play host to the Thunder on Saturday. The Raptors (16-7) take on the Kings in Sacramento on Sunday.

Figure 4.8: Two generated examples from the reconstructed ROTOWIRE (VISION) dataset which extends ROTOWIRE (FULL) with visual inputs.

4.5 Conclusion

By addressing data fidelity in sports game summarization through supplementary input records, we achieve larger performance gains than those proposed in prior purification schemes. Furthermore, sparse attention LLMs generalized well in the large DTG datasets proposed here. When provided sufficient input data, they successfully encoded the linearized input format and produced truthfully grounded and coherent text. For future work that involves even more input records, these models remain a top candidate for appropriate benchmarking.

On the other hand, reformulating the relation evaluation objective in a "text-to-text" fashion and making use of generative LLMs procured models that are easier to process with stronger generalization. For example, there is no need to update the architecture every time we wish to scale the model for additional entities. By using human-verified data and incorporating data augmentation the updated autoeval model(s) are more grounded and reliable.

Finally, adopting the sports summarization dataset to multiple modalities has proven to be feasible and opens a new avenue for expansion. DONUT is not as strong with language intricacies and coherent generation, however that is a model-specific issue that can be easily resolved. The official game documents we used also have comprehensive information about play-by-play descriptors, per-quarter boxscores, etc. that one can take advantage of in future work. One issue is the conflicting layout and font sizes which may need to be normalized in order to work at lower resolutions.

4.5.1 Future Work

The natural followup with sports game summarization is twofold. The first is to determine how to further fill-in the knowledge gap between the input records and summaries. One can look into incorporating more granular features such as play-by-play information, per-quarter boxscores, etc. This imposes subsidiary challenges as the number of tokens and required compute will be astronomical, even with sparse attention. For example, future work will consequently lead to looking at alternate input formats capable of compressing information into shorter sequences (i.e. images, videos, etc.) or combining them into some multi-component step-wise paragraph pipeline.

The second is to extend current methods of validating correctness in the generated summaries. The revised relation extraction approach proposed here is quite effective at

determining grounded correctness, however it is limited to only the entities that are integrated into the autoeval models. Future research would have to look into incorporating strategies of evaluating substrings (i.e. specific plays or events that are phrased differently, domain-specific reasoning, etc.). We can also explore prompting LLM decoders as they will have considerable "knowledge" about this domain context. No matter the approach, it is also crucial to conduct studies to ensure that the metrics and model performance are aligned with human evaluators.

Being consistent with the sports domain, it is possible to create similar datasets for other professional leagues. Most of these other subdomains will naturally pose similar challenges to those found here and can largely take advantage of the same techniques. Outside of game summarization the applicability of statistical learning to sports is almost limitless. Additional problems for example can be catered towards forecasting (i.e. projecting future winners, team/player statistics, injury and health statistics, etc.).

Taking a step back, we can also continue on exploring *foundational* multimodal models suitable for data-to-text generation. From preliminary experiments, current open sourced models for document understanding are not strong enough in language understanding. However, building better models would necessitate sufficient quantities of data to simultaneously grasp both document and language understanding. Future work will revolve around exhaustively constructing new datasets to create visual data-to-text objectives (whether novel or extending prior contributions) for both pre-training and fine-tuning as well as study respective generalization properties.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P. A., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. and Zhang, X. (2016), 'Tensorflow: A system for large-scale machine learning', **abs/1605.08695**.

Ainslie, J., Ontañón, S., Alberti, C., Pham, P., Ravula, A. and Sanghai, S. K. (2020), 'Etc: Encoding long and structured data in transformers', **abs/2004.08483**.

Anselma, L. and Mazzei, A. (2018), Designing and testing the messages produced by a virtual dietitian, *in* 'International Conference on Natural Language Generation'.

Aoki, T., Miyazawa, A., Ishigaki, T., Goshima, K., Aoki, K., Kobayashi, I., Takamura, H. and Miyao, Y. (2018), Generating market comments referring to external resources, *in* 'International Conference on Natural Language Generation'.

Athiwaratkun, B., dos Santos, C. N., Krone, J. and Xiang, B. (2020), Augmented natural language for generative sequence labeling, *in* 'EMNLP'.

Ba, J. L., Kiros, J. R. and Hinton, G. E. (2016), 'Layer normalization'.
URL: *https://arxiv.org/abs/1607.06450*

Bahdanau, D., Cho, K. and Bengio, Y. (2014), 'Neural machine translation by jointly learning to align and translate'.
URL: *https://arxiv.org/abs/1409.0473*

Baker, J. K. (1990), Stochastic modeling for automatic speech understanding.

Banerjee, S. and Lavie, A. (2005), Meteor: An automatic metric for mt evaluation with improved correlation with human judgments, *in* 'IEEvaluation@ACL'.

Bao, H., Dong, L. and Wei, F. (2021), 'Beit: Bert pre-training of image transformers', *ArXiv* **abs/2106.08254**.

Barron, A. R. (1993), 'Universal approximation bounds for superpositions of a sigmoidal function', *IEEE Trans. Inf. Theory* **39**, 930–945.

Barzilay, R. and Lapata, M. (2005), Modeling local coherence: An entity-based approach, *in* 'Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)', Association for Computational Linguistics, pp. 141–148.
URL: *https://aclanthology.org/P05-1018*

Beltagy, I., Peters, M. E. and Cohan, A. (2020), 'Longformer: The long-document transformer', **abs/2004.05150**.

Bengio, Y. (2012), Practical recommendations for gradient-based training of deep architectures, *in* 'Neural Networks: Tricks of the Trade'.

Bengio, Y., Ducharme, R., Vincent, P. and Janvin, C. (2000), A neural probabilistic language model, *in* 'J. Mach. Learn. Res.'.

Bertsekas, D. P. (1995), 'Nonlinear programming', *Journal of the Operational Research Society* **48**, 334.

Bird, S. (2004), 'Nltk: The natural language toolkit', **cs.CL/0205028**.

Black, S., Biderman, S. R., Hallahan, E., Anthony, Q. G., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., Pieler, M. M., Prashanth, U. S., Purohit, S., Reynolds, L., Tow, J., Wang, B. and Weinbach, S. (2022), 'Gpt-neox-20b: An open-source autoregressive language model', *ArXiv* **abs/2204.06745**.

Braun, D., Reiter, E. and Siddharthan, A. (2018), 'Saferdrive: An nlg-based behaviour change support system for drivers', *Natural Language Engineering* **24**, 551 – 588.

Brill, E. and Moore, R. C. (2000), An improved error model for noisy channel spelling correction, *in* 'Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics', Association for Computational Linguistics, pp. 286–293.
URL: *https://aclanthology.org/P00-1037*

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T. J., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C.,

Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. and Amodei, D. (2020), 'Language models are few-shot learners', **abs/2005.14165**.

Bryson, A. E., Ho, Y.-C. and Siouris, G. M. (1969), 'Applied optimal control: Optimization, estimation, and control', *IEEE Transactions on Systems, Man, and Cybernetics* **9**, 366–367.

Buck, C., Heafield, K. and van Ooyen, B. (2014), N-gram counts and language models from the common crawl, *in* 'LREC'.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. and Zagoruyko, S. (2020), 'End-to-end object detection with transformers', *ArXiv* **abs/2005.12872**.

Chang, E., Shen, X., Zhu, D., Demberg, V. and Su, H. (2021), Neural data-to-text generation with lm-based text augmentation, *in* 'EACL'.

Chen, D. L. and Mooney, R. J. (2008), Learning to sportscast: a test of grounded language acquisition, *in* 'International Conference on Machine Learning'.

Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., Ma, S., Xu, C., Xu, C. and Gao, W. (2020), 'Pre-trained image processing transformer', *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 12294–12305.

Chen, J., Tam, D., Raffel, C., Bansal, M. and Yang, D. (2021), 'An empirical survey of data augmentation for limited data learning in nlp', *arXiv preprint arXiv:2106.07499* .

Chen, M., Wiseman, S. and Gimpel, K. (2020), Wikitablet: A large-scale data-to-text dataset for generating wikipedia article sections, *in* 'Findings'.

Chen, S. F. and Goodman, J. (1996), 'An empirical study of smoothing techniques for language modeling', **13**, 359–393.

Cheng, J., Dong, L. and Lapata, M. (2016), 'Long short-term memory-networks for machine reading'.
URL: *https://arxiv.org/abs/1601.06733*

Cho, K., van Merrienboer, B., Çaglar Gülçehre, Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014), Learning phrase representations using rnn encoder–decoder for statistical machine translation, *in* 'EMNLP'.

Choi, S., Hwang, J.-i., Noh, H. and Lee, Y. (2021), 'May the force be with your copy mechanism: Enhanced supervised-copy method for natural language generation'. **URL:** *https://arxiv.org/abs/2112.10360*

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlós, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L. J. and Weller, A. (2020), 'Rethinking attention with performers', *ArXiv* **abs/2009.14794**.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A. B., Barnes, P., Tay, Y., Shazeer, N. M., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B. C., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., García, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E. O., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Díaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K. S., Eck, D., Dean, J., Petrov, S. and Fiedel, N. (2022), 'Palm: Scaling language modeling with pathways', **abs/2204.02311**.

Christiano, P. F., Leike, J., Brown, T. B., Martic, M., Legg, S. and Amodei, D. (2017), 'Deep reinforcement learning from human preferences', *ArXiv* **abs/1706.03741**.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Narang, S., Mishra, G., Yu, A. W., Zhao, V., Huang, Y., Dai, A. M., Yu, H., Petrov, S., Chi, E., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. and Wei, J. (2022), 'Scaling instruction-finetuned language models', *ArXiv* **abs/2210.11416**.

Clevert, D.-A., Unterthiner, T. and Hochreiter, S. (2015), 'Fast and accurate deep network learning by exponential linear units (elus)'.

Cohan, A., Dernoncourt, F., Kim, D. S., Bui, T., Kim, S., Chang, W. and Goharian, N. (2018), A discourse-aware attention model for abstractive summarization of long documents, *in* 'North American Chapter of the Association for Computational Linguistics'.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P. P. (2011), 'Natural language processing (almost) from scratch', *ArXiv* **abs/1103.0398**.

Cramér, H. (1946), *Mathematical Methods of Statistics (PMS-9)*, Princeton University Press.
URL: *http://www.jstor.org/stable/j.ctt1bpm9r4*

Cybenko, G. V. (1989), 'Approximation by superpositions of a sigmoidal function', *Mathematics of Control, Signals and Systems* **2**, 303–314.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009), 'Imagenet: A large-scale hierarchical image database', *2009 IEEE Conference on Computer Vision and Pattern Recognition* pp. 248–255.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019), 'Bert: Pre-training of deep bidirectional transformers for language understanding', **abs/1810.04805**.

dos Santos, C. N., Xiang, B. and Zhou, B. (2015), Classifying relations by ranking with convolutional neural networks, *in* 'ACL'.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N. (2020), 'An image is worth 16x16 words: Transformers for image recognition at scale', *ArXiv* **abs/2010.11929**.

Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q. H., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I. and Florence, P. R. (2023), 'Palm-e: An embodied multimodal language model', *ArXiv* **abs/2303.03378**.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., Zoph, B., Fedus, L., Bosma, M., Zhou, Z., Wang, T., Wang, Y. E., Webster, K., Pellat, M., Robinson, K., Meier-Hellstern, K. S., Duke, T., Dixon, L., Zhang, K., Le, Q. V., Wu, Y., Chen, Z. and Cui, C. (2022), Glam: Efficient scaling of language models with mixture-of-experts, *in* 'ICML'.

Duchi, J. C., Hazan, E. and Singer, Y. (2010), Adaptive subgradient methods for online learning and stochastic optimization, *in* 'J. Mach. Learn. Res.'.

Eisenstein, J. (2018), 'Natural language processing'.
URL: *https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf*

Ethan Joseph, J. L. and Si, M. (2021), 'Improving data-to-text generation via preserving high-frequency phrases and fact-checking'.

Fan, A., Lewis, M. and Dauphin, Y. (2018), Hierarchical neural story generation, *in* 'Annual Meeting of the Association for Computational Linguistics'.

Fang, Y., Liao, B., Wang, X., Fang, J., Qi, J., Wu, R., Niu, J. and Liu, W. (2021), You only look at one sequence: Rethinking transformer in vision through object detection, *in* 'Neural Information Processing Systems'.

Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T. and Hovy, E. (2021), 'A survey of data augmentation approaches for nlp', *arXiv preprint arXiv:2105.03075* .

Freitas, D. D., Luong, M.-T., So, D. R., Hall, J., Fiedel, N., Thoppilan, R., Yang, Z., Kulshreshtha, A., Nemade, G., Lu, Y. and Le, Q. V. (2020), 'Towards a human-like open-domain chatbot', *ArXiv* **abs/2001.09977**.

Fukushima, K. (1969), 'Visual feature extraction by a multilayered network of analog threshold elements', **5**, 322–333.

Gao, L., Biderman, S. R., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S. and Leahy, C. (2021), 'The pile: An 800gb dataset of diverse text for language modeling', **abs/2101.00027**.

Gardent, C., Shimorina, A., Narayan, S. and Perez-Beltrachini, L. (2017), The webnlg challenge: Generating text from rdf data, *in* 'INLG'.

Glorot, X. and Bengio, Y. (2010), Understanding the difficulty of training deep feedforward neural networks, *in* 'AISTATS'.

Glorot, X., Bordes, A. and Bengio, Y. (2011), Deep sparse rectifier neural networks, *in* 'AISTATS'.

Goldberg, Y. (2016), 'A primer on neural network models for natural language processing', *ArXiv* **abs/1510.00726**.

Gong, L., Crego, J. M. and Senellart, J. (2019), Enhanced transformer model for data-to-text generation, *in* 'EMNLP'.

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep Learning*, MIT Press. http://www.deeplearningbook.org.

Goodman, J. (2001), 'A bit of progress in language modeling', **cs.CL/0108005**.

Goyal, P., Dollár, P., Girshick, R. B., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y. and He, K. (2017), 'Accurate, large minibatch sgd: Training imagenet in 1 hour', *ArXiv* **abs/1706.02677**.

Graves, A. (2013), 'Generating sequences with recurrent neural networks'.
URL: *https://arxiv.org/abs/1308.0850*

Graves, A., Wayne, G. and Danihelka, I. (2014), 'Neural turing machines'.
URL: *https://arxiv.org/abs/1410.5401*

Gu, J., Kwon, H., Wang, D., Ye, W., Li, M., Chen, Y.-H., Lai, L., Chandra, V. and Pan, D. Z. (2021), 'Multi-scale high-resolution vision transformer for semantic segmentation', *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 12084–12093.

Gu, J., Lu, Z., Li, H. and Li, V. O. K. (2016), 'Incorporating copying mechanism in sequence-to-sequence learning'.
URL: *https://arxiv.org/abs/1603.06393*

Gulcehre, C., Ahn, S., Nallapati, R., Zhou, B. and Bengio, Y. (2016), 'Pointing the unknown words'.
URL: *https://arxiv.org/abs/1603.08148*

Guo, M., Ainslie, J., Uthus, D. C., Ontañón, S., Ni, J., Sung, Y.-H. and Yang, Y. (2022), Longt5: Efficient text-to-text transformer for long sequences, *in* 'NAACL-HLT'.

Hastie, T., Tibshirani, R. and Friedman, J. (2009), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*.

He, K., Zhang, X., Ren, S. and Sun, J. (2015*a*), 'Deep residual learning for image recognition'.
URL: *https://arxiv.org/abs/1512.03385*

He, K., Zhang, X., Ren, S. and Sun, J. (2015*b*), 'Delving deep into rectifiers: Surpassing human-level performance on imagenet classification', *2015 IEEE International Conference on Computer Vision (ICCV)* pp. 1026–1034.

Hendrycks, D. and Gimpel, K. (2016), 'Gaussian error linear units (gelus)'.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O. and Sifre, L. (2022), 'Training compute-optimal large language models', *ArXiv* **abs/2203.15556**.

Holtzman, A., Buys, J., Forbes, M. and Choi, Y. (2019), 'The curious case of neural text degeneration', *ArXiv* **abs/1904.09751**.

Hornik, K. (1991), 'Approximation capabilities of multilayer feedforward networks', **4**, 251–257.

Hornik, K., Stinchcombe, M. B. and White, H. L. (1989), 'Multilayer feedforward networks are universal approximators', *Neural Networks* **2**, 359–366.

Hornik, K., Stinchcombe, M. B. and White, H. L. (1990), 'Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks', *Neural Networks* **3**, 551–560.

Howard, A. G., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V. and Adam, H. (2019), 'Searching for mobilenetv3', pp. 1314–1324.

Howard, J. and Ruder, S. (2018), Universal language model fine-tuning for text classification, *in* 'ACL'.

Huang, S., Dong, L., Wang, W., Hao, Y., Singhal, S., Ma, S., Lv, T., Cui, L., Mohammed, O. K., Liu, Q., Aggarwal, K., Chi, Z., Bjorck, J., Chaudhary, V., Som, S., Song, X. and Wei, F. (2023), 'Language is not all you need: Aligning perception with language models', *ArXiv* **abs/2302.14045**.

Huang, Y., Cheng, Y., Chen, D., Lee, H., Ngiam, J., Le, Q. V. and Chen, Z. (2019), 'Gpipe: Efficient training of giant neural networks using pipeline parallelism', **abs/1811.06965**.

Huang, Y., Lv, T., Cui, L., Lu, Y. and Wei, F. (2022), 'Layoutlmv3: Pre-training for document ai with unified text and image masking', *Proceedings of the 30th ACM International Conference on Multimedia* .

Ioffe, S. and Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* 'ICML'.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013), *An Introduction to Statistical Learning: with Applications in R*, Springer.
URL: *https://faculty.marshall.usc.edu/gareth-james/ISL/*

Jarrett, K., Kavukcuoglu, K., Ranzato, M. and LeCun, Y. (2009), 'What is the best multi-stage architecture for object recognition?', pp. 2146–2153.

Jelinek, F. (1976), 'Continuous speech recognition by statistical methods', **64**, 532–556.

Jelinek, F. (1980), Interpolated estimation of markov source parameters from sparse data.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L. and Levy, O. (2020), 'Spanbert: Improving pre-training by representing and predicting spans', **8**, 64–77.

Jurafsky, D. and Martin, J. H. (2022), Speech and language processing (3rd ed.).

Juraska, J., Bowden, K. K. and Walker, M. A. (2019), 'Viggo: A video game corpus for data-to-text generation in open-domain conversation', *ArXiv* **abs/1910.12129**.

Kale, M. (2020), 'Text-to-text pre-training for data-to-text tasks', **abs/2005.10433**.

Kanthara, S., Leong, R. T. K., Lin, X., Masry, A., Thakkar, M., Hoque, E. and Joty, S. R. (2022), Chart-to-text: A large-scale benchmark for chart summarization, *in* 'Annual Meeting of the Association for Computational Linguistics'.

Kasner, Z. and Dusek, O. (2022), 'Neural pipeline for zero-shot data-to-text generation', **abs/2203.16279**.

Katharopoulos, A., Vyas, A., Pappas, N. and Fleuret, F. (2020), 'Transformers are rnns: Fast autoregressive transformers with linear attention', *ArXiv* **abs/2006.16236**.

Keymanesh, M., Benton, A. and Dredze, M. (2022), 'What makes data-to-text generation hard for pretrained language models?', **abs/2205.11505**.

Kim, G., Hong, T., Yim, M., Nam, J., Park, J., Yim, J., Hwang, W., Yun, S., Han, D. and Park, S. (2022), Ocr-free document understanding transformer, *in* 'European Conference on Computer Vision'.

Kingma, D. P. and Ba, J. (2015), 'Adam: A method for stochastic optimization', **abs/1412.6980**.

Klambauer, G., Unterthiner, T., Mayr, A. and Hochreiter, S. (2017), 'Self-normalizing neural networks', **abs/1706.02515**.

Klein, G., Kim, Y., Deng, Y., Senellart, J. and Rush, A. M. (2017), 'Opennmt: Open-source toolkit for neural machine translation', *ArXiv* **abs/1701.02810**.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S. and Houlsby, N. (2019*a*), Big transfer (bit): General visual representation learning, *in* 'European Conference on Computer Vision'.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S. and Houlsby, N. (2019*b*), 'Large scale learning of general visual representations for transfer', **abs/1912.11370**.

Koncel-Kedziorski, R., Bekal, D., Luan, Y., Lapata, M. and Hajishirzi, H. (2019), Text generation from knowledge graphs with graph transformers, *in* 'North American Chapter of the Association for Computational Linguistics'.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), 'Imagenet classification with deep convolutional neural networks', *Communications of the ACM* **60**, 84 – 90.

Kudo, T. (2018), Subword regularization: Improving neural network translation models with multiple subword candidates, *in* 'ACL'.

Kudo, T. and Richardson, J. (2018), Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, *in* 'EMNLP'.

Kulikov, I., Miller, A. H., Cho, K. and Weston, J. (2018), 'Importance of search and evaluation strategies in neural dialogue modeling'.

Lebret, R., Grangier, D. and Auli, M. (2016), Neural text generation from structured data with application to the biography domain, *in* 'Conference on Empirical Methods in Natural Language Processing'.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E. and Jackel, L. D. (1989), 'Backpropagation applied to handwritten zip code recognition', *Neural Computation* **1**, 541–551.

Lewis, D. D., Agam, G., Argamon, S. E., Frieder, O., Grossman, D. A. and Heard, J. (2006), 'Building a test collection for complex document information processing', *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* .

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. and Zettlemoyer, L. (2019), 'Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension'.
URL: *https://arxiv.org/abs/1910.13461*

Lewkowycz, A., Andreassen, A. J., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V. V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., Wu, Y., Neyshabur, B., Gur-Ari, G. and Misra, V. (2022), 'Solving quantitative reasoning problems with language models', **abs/2206.14858**.

Li, H., Xu, Z., Taylor, G. and Goldstein, T. (2017), Visualizing the loss landscape of neural nets, *in* 'Neural Information Processing Systems'.

Li, M., Lv, T., Cui, L., Lu, Y., Florêncio, D. A. F., Zhang, C., Li, Z. and Wei, F. (2021), 'Trocr: Transformer-based optical character recognition with pre-trained models', *ArXiv* **abs/2109.10282**.

Liang, P., Jordan, M. I. and Klein, D. (2009), Learning semantic correspondences with less supervision, *in* 'Annual Meeting of the Association for Computational Linguistics'.

Lin, C.-Y. (2004), Rouge: A package for automatic evaluation of summaries, *in* 'ACL 2004'.

Lin, C.-Y. and Hovy, E. H. (2003), Automatic evaluation of summaries using n-gram co-occurrence statistics, *in* 'NAACL'.

Lin, C.-Y. and Och, F. J. (2004), Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics, *in* 'ACL'.

Liu, L., Liu, X., Gao, J., Chen, W. and Han, J. (2020), 'Understanding the difficulty of training transformers', *ArXiv* **abs/2004.08249**.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L. and Shazeer, N. M. (2018), 'Generating wikipedia by summarizing long sequences', *ArXiv* **abs/1801.10198**.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019), 'Roberta: A robustly optimized bert pretraining approach'.
URL: *https://arxiv.org/abs/1907.11692*

Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., Wei, F. and Guo, B. (2021), 'Swin transformer v2: Scaling up capacity and resolution',

2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 11999–12009.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. and Guo, B. (2021), 'Swin transformer: Hierarchical vision transformer using shifted windows', *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* pp. 9992–10002.

Lu, Z., Pu, H., Wang, F., Hu, Z. and Wang, L. (2017), The expressive power of neural networks: A view from the width, *in* 'NIPS'.

Luong, M.-T., Le, Q. V., Sutskever, I., Vinyals, O. and Kaiser, L. (2015), 'Multi-task sequence to sequence learning'.
URL: *https://arxiv.org/abs/1511.06114*

Luong, M.-T., Pham, H. and Manning, C. D. (2015), 'Effective approaches to attention-based neural machine translation'.
URL: *https://arxiv.org/abs/1508.04025*

Maas, A. L. (2013), Rectifier nonlinearities improve neural network acoustic models.

Mahajan, D. K., Girshick, R. B., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A. and van der Maaten, L. (2018), 'Exploring the limits of weakly supervised pretraining', *ArXiv* **abs/1805.00932**.

McClelland, J. L., Rumelhart, D. E. and Hinton, G. E. (1986), The appeal of parallel distributed processing.

McCloskey, M. and Cohen, N. J. (1989), 'Catastrophic interference in connectionist networks: The sequential learning problem', **24**, 109–165.

Mehta, S. V., Rao, J., Tay, Y., Kale, M., Parikh, A. P., Zhong, H. and Strubell, E. (2022), Improving compositional generalization with self-training for data-to-text generation, *in* 'ACL'.

Miller, G. A. (1992), 'Wordnet: A lexical database for english', *Commun. ACM* **38**, 39–41.

Minsky, M. and Papert, S. A. (1969), *Perceptrons: An Introduction to Computational Geometry*, The MIT Press.

Moosavi, N. S., Ruckl'e, A., Roth, D. and Gurevych, I. (2021), 'Learning to reason for text generation from scientific tables', *ArXiv* **abs/2104.08296**.

Murakami, S., Watanabe, A., Miyazawa, A., Goshima, K., Yanase, T., Takamura, H. and Miyao, Y. (2017), Learning to generate market comments from stock prices, *in* 'Annual Meeting of the Association for Computational Linguistics'.

Murphy, K. P. (2022), *Probabilistic Machine Learning: An introduction*, MIT Press.
URL: *probml.ai*

Nair, V. and Hinton, G. E. (2010), Rectified linear units improve restricted boltzmann machines, *in* 'International Conference on Machine Learning'.

Narayan, S., Maynez, J., Adamek, J., Pighin, D., Bratanivc, B. and McDonald, R. T. (2020), Stepwise extractive summarization and planning with structured transformers, *in* 'EMNLP'.

Nash, C., Menick, J., Dieleman, S. and Battaglia, P. W. (2021), 'Generating images with sparse representations', *ArXiv* **abs/2103.03841**.

Nesterov, Y. (2004), Introductory lectures on convex optimization - a basic course, *in* 'Applied Optimization'.

Novikoff, A. B. (1962), On convergence proofs on perceptrons, *in* 'Proceedings of the Symposium on the Mathematical Theory of Automata', Vol. 12, Polytechnic Institute of Brooklyn, pp. 615–622.

Novikova, J., Dusek, O., Curry, A. C. and Rieser, V. (2017), Why we need new evaluation metrics for nlg, *in* 'EMNLP'.

Novikova, J., Dusek, O. and Rieser, V. (2017), The e2e dataset: New challenges for end-to-end generation, *in* 'SIGDIAL Conference'.

Obeid, J. and Hoque, E. (2020), Chart-to-text: Generating natural language descriptions for charts by adapting the transformer model, *in* 'International Conference on Natural Language Generation'.

OpenAI (2023), 'Gpt-4 technical report'.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L. E., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J. and Lowe, R. J. (2022), 'Training language models to follow instructions with human feedback', *ArXiv* **abs/2203.02155**.

Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J. (2002), Bleu: a method for automatic evaluation of machine translation, *in* 'ACL'.

Parikh, A. P., Wang, X., Gehrmann, S., Faruqui, M., Dhingra, B., Yang, D. and Das, D. (2020), 'Totto: A controlled table-to-text generation dataset', **abs/2004.14373**.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S. (2019), Pytorch: An imperative style, high-performance deep learning library, *in* 'NeurIPS'.

Paulus, R., Xiong, C. and Socher, R. (2017), 'A deep reinforced model for abstractive summarization', *ArXiv* **abs/1705.04304**.

Paulus, R., Xiong, C. and Socher, R. (2018), 'A deep reinforced model for abstractive summarization', *ArXiv* **abs/1705.04304**.

Pauws, S. C., Gatt, A., Krahmer, E. J. and Reiter, E. (2018), 'Making effective use of healthcare data using data-to-text technology', *ArXiv* **abs/1808.03507**.

Peters, M. E., Ammar, W., Bhagavatula, C. and Power, R. (2017), 'Semi-supervised sequence tagging with bidirectional language models', *ArXiv* **abs/1705.00108**.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. (2018), Deep contextualized word representations, *in* 'NAACL'.

Phang, J., Zhao, Y. and Liu, P. J. (2022), 'Investigating efficiently extending transformers for long input summarization', *ArXiv* **abs/2208.04347**.

Portet, F., Reiter, E., Hunter, J. and Sripada, S. G. (2007), Automatic generation of textual summaries from neonatal intensive care data, *in* 'Conference on Artificial Intelligence in Medicine in Europe'.

Prince, S. J. (2023), *Understanding Deep Learning*, MIT Press.
URL: *https://udlbook.github.io/udlbook/*

Puduppully, R., Dong, L. and Lapata, M. (2019*a*), 'Data-to-text generation with content selection and planning'.

Puduppully, R., Dong, L. and Lapata, M. (2019*b*), 'Data-to-text generation with entity modeling'.
URL: *https://arxiv.org/abs/1906.03221*

Puduppully, R., Fu, Y. and Lapata, M. (2022), 'Data-to-text generation with variational sequential planning', *Transactions of the Association for Computational Linguistics* **10**, 697–715.

Puduppully, R. and Lapata, M. (2021), 'Data-to-text generation with macro planning'. **URL:** *https://arxiv.org/abs/2102.02723*

Radev, D. R., Zhang, R., Rau, A., Sivaprasad, A., Hsieh, C.-H., Rajani, N., Tang, X., Vyas, A., Verma, N., Krishna, P., Liu, Y., Irwanto, N., Pan, J., Rahman, F., Zaidi, A. Z., Mutuma, M., Tarabar, Y., Gupta, A., Yu, T., Tan, Y. C., Lin, X. V., Xiong, C. and Socher, R. (2020), Dart: Open-domain structured data record to text generation, *in* 'North American Chapter of the Association for Computational Linguistics'.

Radford, A. and Narasimhan, K. (2018), Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019), Language models are unsupervised multitask learners.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., Rutherford, E., Hennigan, T., Menick, J., Cassirer, A., Powell, R., van den Driessche, G., Hendricks, L. A., Rauh, M., Huang, P.-S., Glaese, A., Welbl, J., Dathathri, S., Huang, S., Uesato, J., Mellor, J. F. J., Higgins, I., Creswell, A., McAleese, N., Wu, A., Elsen, E., Jayakumar, S. M., Buchatskaya, E., Budden, D., Sutherland, E., Simonyan, K., Paganini, M., Sifre, L., Martens, L., Li, X. L., Kuncoro, A., Nematzadeh, A., Gribovskaya, E., Donato, D., Lazaridou, A., Mensch, A., Lespiau, J.-B., Tsimpoukelli, M., Grigorev, N. K., Fritz, D., Sottiaux, T., Pajarskas, M., Pohlen, T., Gong, Z., Toyama, D., de Masson d'Autume, C., Li, Y., Terzi, T., Mikulik, V., Babuschkin, I., Clark, A., de Las Casas, D., Guy, A., Jones, C., Bradbury, J., Johnson, M. G., Hechtman, B. A., Weidinger, L., Gabriel, I., Isaac, W. S., Lockhart, E., Osindero, S., Rimell, L., Dyer, C., Vinyals, O., Ayoub, K. W., Stanway, J., Bennett, L. L., Hassabis, D., Kavukcuoglu, K. and Irving, G. (2021), 'Scaling language models: Methods, analysis & insights from training gopher', **abs/2112.11446**.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P. J. (2019), 'Exploring the limits of transfer learning with a unified text-to-text transformer'.
URL: *https://arxiv.org/abs/1910.10683*

Ramachandran, P., Zoph, B. and Le, Q. V. (2018), 'Searching for activation functions', **abs/1710.05941**.

Rao, C. R. (1945), Information and the accuracy attainable in the estimation of statistical parameters.

Rebuffel, C., Soulier, L., Scoutheeten, G. and Gallinari, P. (2019), 'A hierarchical model for data-to-text generation'.
URL: *https://arxiv.org/abs/1912.10011*

Rei, R., Stewart, C. A., Farinha, A. C. and Lavie, A. (2020), Comet: A neural framework for mt evaluation, *in* 'EMNLP'.

Reiter, E. and Dale, R. (1997), 'Building applied natural language generation systems', *Natural Language Engineering* **3**, 57 – 87.

Robin, J. (1994), Revision-based generation of natural language summaries providing historical background: corpus-based analysis, design, implementation and evaluation.

Rosenblatt, F. (1958), 'The perceptron: a probabilistic model for information storage and organization in the brain.', **65 6**, 386–408.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986*a*), Learning internal representations by error propagation.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986*b*), 'Learning representations by back-propagating errors', *Nature* **323**, 533–536.

Rumelhart, D. E. and McClelland, J. L. (1986), Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations.

Schuhmann, C., Vencu, R., Beaumont, R., Kaczmarczyk, R., Mullis, C., Katta, A., Coombes, T., Jitsev, J. and Komatsuzaki, A. (2021), 'Laion-400m: Open dataset of clip-filtered 400 million image-text pairs', *ArXiv* **abs/2111.02114**.

Schuster, M. and Nakajima, K. (2012), 'Japanese and korean voice search', pp. 5149–5152.

Schuster, M. and Paliwal, K. K. (1997), 'Bidirectional recurrent neural networks', *IEEE Trans. Signal Process.* **45**, 2673–2681.

Sellam, T., Das, D. and Parikh, A. P. (2020), Bleurt: Learning robust metrics for text generation, *in* 'ACL'.

Sennrich, R., Haddow, B. and Birch, A. (2016), 'Neural machine translation of rare words with subword units', **abs/1508.07909**.

Shang, W., Sohn, K., Almeida, D. and Lee, H. (2016), 'Understanding and improving convolutional neural networks via concatenated rectified linear units', **abs/1603.05201**.

Sharma, M., Brownstein, J. S. and Ramakrishnan, N. (2021), 'T3: Domain-agnostic neural time-series narration', *2021 IEEE International Conference on Data Mining (ICDM)* pp. 1324–1329.

Sharma, M., Gogineni, A. K. and Ramakrishnan, N. (2022), 'Innovations in neural data-to-text generation', *ArXiv* **abs/2207.12571**.

Shaw, P., Uszkoreit, J. and Vaswani, A. (2018), Self-attention with relative position representations, *in* 'NAACL'.

Shazeer, N. M. and Stern, M. (2018), 'Adafactor: Adaptive learning rates with sublinear memory cost', **abs/1804.04235**.

Shorten, C., Khoshgoftaar, T. M. and Furht, B. (2021), 'Text data augmentation for deep learning', *Journal of big Data* **8**(1), 1–34.

Smith, L. N. (2018), 'A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay', *ArXiv* **abs/1803.09820**.

Smith, S., Patwary, M. A., Norick, B., LeGresley, P., Rajbhandari, S., Casper, J., Liu, Z., Prabhumoye, S., Zerveas, G., Korthikanti, V. A., Zhang, E., Child, R., Aminabadi, R. Y., Bernauer, J., Song, X., Shoeybi, M., He, Y., Houston, M., Tiwary, S. and Catanzaro, B. (2022), 'Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model', **abs/2201.11990**.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014), 'Dropout: a simple way to prevent neural networks from overfitting', **15**, 1929–1958.

Su, Y., Meng, Z., Baker, S. and Collier, N. (2021), Few-shot table-to-text generation with prototype memory, *in* 'EMNLP'.

Sun, C., Shrivastava, A., Singh, S. and Gupta, A. K. (2017), 'Revisiting unreasonable effectiveness of data in deep learning era', *2017 IEEE International Conference on Computer Vision (ICCV)* pp. 843–852.

Sutskever, I., Vinyals, O. and Le, Q. V. (2014), Sequence to sequence learning with neural networks, NIPS'14, MIT Press, Cambridge, MA, USA, p. 3104–3112.

Tanaka-Ishii, K., Hasida, K. and Noda, I. (1998), Reactive content selection in the generation of real-time soccer commentary, *in* 'COLING 1998 Volume 2: The 17th International Conference on Computational Linguistics'.
URL: *https://aclanthology.org/C98-2204*

Tay, Y., Dehghani, M., Tran, V. Q., García, X., Bahri, D., Schuster, T., Zheng, H., Houlsby, N. and Metzler, D. (2022), 'Unifying language learning paradigms', *ArXiv* **abs/2205.05131**.

Thoppilan, R., Freitas, D. D., Hall, J., Shazeer, N. M., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L., Du, Y., Li, Y., Lee, H., Zheng, H., Ghafouri, A., Menegali, M., Huang, Y., Krikun, M., Lepikhin, D., Qin, J., Chen, D., Xu, Y., Chen, Z., Roberts, A., Bosma, M., Zhou, Y., Chang, C.-C., Krivokon, I. A., Rusch, W. J., Pickett, M., Meier-Hellstern, K. S., Morris, M. R., Doshi, T., Santos, R. D., Duke, T., Søraker, J. H., Zevenbergen, B., Prabhakaran, V., Díaz, M., Hutchinson, B., Olson, K., Molina, A., Hoffman-John, E., Lee, J., Aroyo, L., Rajakumar, R., Butryna, A., Lamm, M., Kuzmina, V. O., Fenton, J., Cohen, A., Bernstein, R., Kurzweil, R., Aguera-Arcas, B., Cui, C., Croak, M., Chi, E. and Le, Q. (2022), 'Lamda: Language models for dialog applications', *ArXiv* **abs/2201.08239**.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the royal statistical society series b-methodological* **58**, 267–288.

Tieleman, T. and Hinton, G. (2012), 'Divide the gradient by a running average of its recent magnitude', pp. 4, 26–31.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A. and J'egou, H. (2020), Training data-efficient image transformers & distillation through attention, *in* 'International Conference on Machine Learning'.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E. and Lample, G. (2023), 'Llama: Open and efficient foundation language models', *ArXiv* **abs/2302.13971**.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2017), 'Attention is all you need'.
URL: *https://arxiv.org/abs/1706.03762*

Vijayakumar, A. K., Cogswell, M., Selvaraju, R. R., Sun, Q., Lee, S., Crandall, D. J. and Batra, D. (2016), 'Diverse beam search: Decoding diverse solutions from neural sequence models', *ArXiv* **abs/1610.02424**.

Vinyals, O., Fortunato, M. and Jaitly, N. (2015), 'Pointer networks'.
URL: *https://arxiv.org/abs/1506.03134*

Wang, H. (2019), Revisiting challenges in data-to-text generation with fact grounding, *in* 'Proceedings of the 12th International Conference on Natural Language Generation', Association for Computational Linguistics, pp. 311–322.
URL: *https://aclanthology.org/W19-8639*

Wang, J., Yang, Z., Hu, X., Li, L., Lin, K., Gan, Z., Liu, Z., Liu, C. and Wang, L. (2022), 'Git: A generative image-to-text transformer for vision and language', *arXiv preprint arXiv:2205.14100* .

Wang, S., Li, B. Z., Khabsa, M., Fang, H. and Ma, H. (2020), 'Linformer: Self-attention with linear complexity', *ArXiv* **abs/2006.04768**.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M. and Le, Q. V. (2022), 'Finetuned language models are zero-shot learners', *ArXiv* **abs/2109.01652**.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., hsin Chi, E. H., Le, Q. and Zhou, D. (2022), 'Chain of thought prompting elicits reasoning in large language models', *ArXiv* **abs/2201.11903**.

Wei, J. and Zou, K. (2019), 'Eda: Easy data augmentation techniques for boosting performance on text classification tasks', *arXiv preprint arXiv:1901.11196* .

Werbos, P. J. (1974), Beyond regression : "new tools for prediction and analysis in the behavioral sciences.

Wiseman, S., Shieber, S. and Rush, A. (2017), Challenges in data-to-document generation, *in* 'Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing', Association for Computational Linguistics, pp. 2253–2263.
URL: *https://aclanthology.org/D17-1239*

Wolpert, D. H. (1996), 'The lack of a priori distinctions between learning algorithms', *Neural Computation* **8**, 1341–1390.

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Álvarez, J. M. and Luo, P. (2021), Segformer: Simple and efficient design for semantic segmentation with transformers, *in* 'Neural Information Processing Systems'.

Xie, Q., Hovy, E. H., Luong, M.-T. and Le, Q. V. (2019), 'Self-training with noisy student improves imagenet classification', *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 10684–10695.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L. and Liu, T.-Y. (2020), On layer normalization in the transformer architecture, *in* 'International Conference on Machine Learning'.

Xu, Y., Li, M., Cui, L., Huang, S., Wei, F. and Zhou, M. (2019), 'Layoutlm: Pre-training of text and layout for document image understanding', *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* .

Xu, Y., Lv, T., Cui, L., Wang, G., Lu, Y., Florêncio, D. A. F., Zhang, C. and Wei, F. (2021), 'Layoutxlm: Multimodal pre-training for multilingual visually-rich document understanding', *ArXiv* **abs/2104.08836**.

Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., Lu, Y., Florêncio, D. A. F., Zhang, C., Che, W., Zhang, M. and Zhou, L. (2020), 'Layoutlmv2: Multi-modal pre-training for visually-rich document understanding', *ArXiv* **abs/2012.14740**.

Yang, Z., Blunsom, P., Dyer, C. and Ling, W. (2016), 'Reference-aware language models'. **URL:** *https://arxiv.org/abs/1611.01628*

Ye, L., Rochan, M., Liu, Z. and Wang, Y. (2019), 'Cross-modal self-attention network for referring image segmentation', *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 10494–10503.

Yermakov, R., Drago, N. and Ziletti, A. (2021), Biomedical data-to-text generation via fine-tuning transformers, *in* 'INLG'.

Yim, M., Kim, Y., Cho, H.-C. and Park, S. (2021), Synthtiger: Synthetic text image generator towards better text recognition models, *in* 'IEEE International Conference on Document Analysis and Recognition'.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontañón, S., Pham, P., Ravula, A., Wang, Q., Yang, L. and Ahmed, A. (2020), 'Big bird: Transformers for longer sequences', **abs/2007.14062**.

Zeiler, M. D. (2012), 'Adadelta: An adaptive learning rate method', **abs/1212.5701**.

Zeng, D., Liu, K., Lai, S., Zhou, G. and Zhao, J. (2014), Relation classification via convolutional deep neural network, *in* 'COLING'.

Zhang, A., Lipton, Z. C., Li, M. and Smola, A. J. (2021), 'Dive into deep learning'.

Zhang, J., Zhao, Y., Saleh, M. and Liu, P. J. (2019), 'Pegasus: Pre-training with extracted gap-sentences for abstractive summarization'.
URL: *https://arxiv.org/abs/1912.08777*

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T. and Zettlemoyer, L. (2022), 'Opt: Open pre-trained transformer language models', *ArXiv* **abs/2205.01068**.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q. and Artzi, Y. (2020), 'Bertscore: Evaluating text generation with bert', **abs/1904.09675**.

Zhang, X., Wei, F. and Zhou, M. (2019), 'Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization', *ArXiv* **abs/1905.06566**.

Zhang, Z. (2004), Weakly-supervised relation classification for information extraction, *in* 'CIKM '04'.

Zhou, G., Li, J., Qian, L. and Zhu, Q. (2008), Semi-supervised learning for relation extraction, *in* 'IJCNLP'.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X. and Dai, J. (2020), 'Deformable detr: Deformable transformers for end-to-end object detection', *ArXiv* **abs/2010.04159**.