UNSUPERVISED PARAPHRASE GENERATION FROM HIERARCHICAL LANGUAGE MODELS

by

Michael Traynor

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2018

to Cub

Table of Contents

List of Tables

List of Figures

Abstract

Paraphrase generation is a challenging problem that requires a semantic representation of language. Language models implemented with deep neural networks (DNN) have the ability to transform text to a real valued vector space that can capture useful semantic information. In light of this, this work employs hierarchical language modeling to produce semantic representations of sentences. An encoder-decoder model is employed that uses four components: a word encoder, sentence encoder, sentence decoder, and word decoder. These components hierarchically convert a sentence from characters through word representations to a fixed-size sentence representation, then back down through words to characters. Many types of neural network are suitable for each component, and a number of them are compared in this work, including a novel architecture, the Self Attentive Recurrent Array (SARAh). The SARAh is shown to perform at least as well as Gated Recurrent Units (GRU) and Transformers on language modeling tasks, and requires fewer parameters. These language models are trained on a large and diverse dataset, but this work also shows that it is possible to fine tune such models to a particular domain, such as the works of a single author. These fine tuned models are able to leverage information learned on the larger dataset in order to perform better on the target domain. Finally, a language model is trained to produce semantic representations of sentences that are subsequently used to produce paraphrases in a completely unsupervised setting. The language model, which is trained to predict the sentence most likely to follow the input sentence, is fine tuned to instead autoencode the input sentence. Given that the sentence encoder produces a semantic representation, it is possible to use a number of techniques to encourage the decoder to generate a paraphrase rather than reconstruct the exact input sentence. These techniques include adding noise to the sentence representation, and sampling characters from the model's output layer.

Acknowledgements

I would like to thank my partner, Sarah, for the endless love and support, the friendships and the meals, our home, and for not letting me lose sight of what's important. I would also like to thank my parents, Liz Rainer and Pat Traynor, for encouraging me to dream, but also for being there whenever I need a place to fall. I would also like to acknowledge my supervisor, Thomas Trappenberg, who opened my eyes to this incredible field of study. Thank you also the readers, Sageev Oore and Robert Beiko for the valuable feedback. I must also thank Dean Tsaltas and everyone at QRA, for allowing and encouraging me to spend so much of my time on learning and developing new skills.

Chapter 1

Introduction

1.1 Overview

This work investigates deep learning solutions to language modeling, and applies hierarchical language modeling as a pre-training phase for the difficult task of paraphrase generation in an unsupervised framework. The hierarchical encoder-decoder model presented explicitly represents three structural levels of text: characters, words, and sentences. The model consists of four main components: the word encoder, sentence encoder, sentence decoder, and word decoder. A variety of implementations for these components are compared based on their language modeling performance, including a novel recurrent neural network architecture, the Self Attentive Recurrent Array (SARAh). The SARAh is shown to perform at least as well as popular, state of the art architectures such as Gated Recurrent Units (GRU)[8] or Transformers[43], and does so with fewer parameters.

The language models in this work are trained on datasets with a diverse set of authors, such as the Gutenberg dataset [23] or the Toronto Books Corpus [52]. However, it may be desirable to produce a language model that is conditioned to a particular linguistic style, such as the style of a particular author. However, it is unlikely that the writings of a single author constitute a large enough corpus on which to train a state of the art language model, which usually requires training sets with hundreds of millions of words [19]. This work shows that a model trained on a diverse dataset can be fine tuned to a particular author, and that a fine tuned model performs better than one trained from scratch on a single author.

Training a large model on these datasets with available hardware takes weeks. In order to allow for the many runs needed to make comparisons between various architectures and amounts of fine tuning, the size of the models is restricted and only a subset of the data is used. While this obviously restricts performance, each architecture is equally limited, and comparisons between them are still valid. The

best performing models are then scaled up and trained on the full corpus in order to assess their ability to capture semantic information an produce paraphrases.

The language models trained in this work are capable of mapping the character representations at the inputs and outputs to internal representations that capture semantic information. The word representations are explored through t-SNE [42] visualisations, which reveal that the depth of the word encoder plays an important part in the ability to transform a word from characters to a semantically relevant space. A deeper word encoder produces better representations than a shallow one, even if the shallow one has large state spaces and more parameters. However, visualisations of the sentence level representations show that later layers, those in the sentence encoder, are able to compensate for a shallow word encoder and embed sentences in a semantic space. It is also shown that sentence level representations can be improved by adding noise to them during training.

A number of recent works have shown that language modeling is a valid strategy for pretraining neural networks, allowing them to learn generalised representations of language that can be used for a number of target tasks. [35] [9] These models encode a sentence into a variable length list of context aware word representations. Allowing a variable length sentence representation is valuable for long sentences, and for tasks such as translation where attention can be applied to the input sequence at each time step [8]. In this work, however, a fixed length sentence representation is desired, because the goal is to decouple the meaning of the sentence from the specific words. This fixed length representation can then be decoded into a paraphrase of the input sentence, without requiring any examples of paraphrases.

Given this unsupervised context, the proposed approach is to pre-train a sentence encoder using language modeling, then freeze its weights and fine tune the decoder to reproduce the input sentence. If the decoder has access to representations of the input words, it could easily learn to simply copy them through and produce an exact replica of the input sentence. Instead, the sentence encoder in this work produces a fixed length vector, and the decoder has access only to this fixed length representation. The model is therefore encouraged to pack any useful information about the sentence into this vector, and does so by learning a semantic embedding space for sentences. During the autoencoding phase, the decoder then learns to reproduce the input sentence from

this semantic embedding space.

In some cases, this is enough to force the decoder to produce a paraphrase. In many cases, the sentence is too long or otherwise too challenging, and the decoder is unsuccessful at producing a paraphrase. In other cases, the decoder may be too successful at reproducing the input sentence word for word. In these cases, it is possible to alter the generated sentence by perturbing the sentence representation with sampled noise. Another technique is to take advantage of the probabilistic interpretation of the model's outputs, which represent a distribution over possible values for the next character. In some cases, sampling from this distribution rather than selecting only the most likely character can force the decoder to produce a paraphrase rather than an exact copy of its inputs.

One of the issues with these non-deterministic approaches to discouraging the decoder from exactly reproducing the input sentence is that success is then also variable. A proposed approach to mitigating this problem is to run the paraphrase generator multiple times on a given sentence, sampling the perturbation and/or the outputs differently each time. This produces a list of candidate paraphrases, from which the best one must be chosen. This can be done manually, but a proposed approach to doing so automatically is to pass the candidates into the encoder and choose the one where the embedded representation is closest to the original sentence. Examples of this approach are shown using a simple Euclidean distance.

A corollary of using this metric to select the best output from a list of candidates is that it is also a useful metric for the quality of the paraphrase, but it should be noted that a distance of 0 indicates that the sentence was simply copied verbatim. Also, it possible to change the meaning of a sentence completely by changing a single word, or sometimes even a single character, such as a comma. This can be problematic unless the model has learned to capture these subtleties in the embedding space in such a way that they are reflected by the distance metric used.

Other works on paraphrase generation have used lexical metrics such as BLEU[33], or ROUGE[28] scores as an approximation for paraphrase quality, but also include human evaluation because these metrics fail to capture the semantic relationships between phrases. Instead they rely on the lexical similarity between generated phrases and a list of reference examples, and therefore fail to make any assessment based on

meaning. For example, without an appropriate reference for a good paraphrase, the sentences *"This is my favourite song"* and *"I love this tune the most"* would be ranked poorly by BLEU. The embedding space metric used in this work, which is related to previous work that makes comparisons based on combinations of word representations [34] [37], is better equipped to handle these semantic relationships.

A major distinction between this and previous works on paraphrase generation is that previous works include at least some supervised training, where explicit examples of paraphrases are provided not just for evaluation but also as a source of the training signal. [34] [27] [49] [14] The models in this work are never given an example of a paraphrase. Although this is a clear opportunity for improvement, the major goal of this work is to assess how well language modeling by itself is able to produce and decode semantic representations. Popular data sources for paraphrase examples include the Quora Question pairs, the Twitter paraphrasing corpus[25], and the Microsoft COCO dataset[29].

1.2 A note on notation

A consistent notation scheme is used in order to enhance the clarity of equations and diagrams in this work. All functions will be denoted by the f symbol, and discriminated by a label placed as a suffix, as in f^{label}. Deep learning involves frequent manipulation of tensors, which will be denoted by bold lower case letters such as $\boldsymbol{v}$. Tensors will be discriminated both by the letter used, as well as a possible label placed in the suffix, such as $\boldsymbol{v}^{(label)}$. The rank of the tensor will be made clear from the indexing paradigm. Each dimension of the tensor is signified by an index present in the suffix. The index will either be a scalar integer or the special $*$ character. An integer index specifies that a slice is being selected from the given dimension, at the index position. For example, the notation $\boldsymbol{a}_2$ signifies a vector (1D tensor) from which we are selecting the second element. An index labeled with the $*$ symbol indicates that all elements from the given dimension are selected. For example, $\boldsymbol{a}_{***}$ indicates a 3D tensor, whereas $\boldsymbol{a}_{1**}$ indicates the 2D tensor slice extracted from position 1 of the first dimension of $\boldsymbol{a}_{***}$. Some more examples are provided below.

- $\boldsymbol{a}$: a scalar

- $\boldsymbol{a}_*$: a vector, unrelated to the scalar $\boldsymbol{a}$
- $\boldsymbol{a}_*^{(label)}$: a vector, labeled to distinguish it from $\boldsymbol{a}_*$
- $\boldsymbol{b}_{**}$: a 2D tensor
- $\boldsymbol{b}_{*2}$: a 1D tensor corresponding to the 2^{nd} column of $\boldsymbol{b}_{**}$
- $\boldsymbol{c}_{342}$: a scalar, selected from the 3D tensor $\boldsymbol{c}_{***}$
- $\boldsymbol{d}_d$: a scalar, selected at the variable index $\boldsymbol{d}$ of vector $\boldsymbol{d}_*$

1.3 Language Modeling

The task in language modeling is to assign probabilities to sentences or sequences of text. That is, we want to model the probability of the next token in a sequence conditional on the ordered sequence of all previous tokens. These tokens may be characters, words, n-grams, or some other tokenized representation. In a character model, for example, the probability of the next character, $\boldsymbol{c}_t$, is modeled as:

$$p(\boldsymbol{c}_t = \boldsymbol{i} | \boldsymbol{c}_{t-1}, \boldsymbol{c}_{t-2}, ..., \boldsymbol{c}_0)$$

Language modeling has many advantages as a metric for the quality of a model's linguistic representations. The task is simple to formulate and easy to implement. It's a semi-supervised task trained on text data, of which there are very large datasets available from a variety of sources such as the web, news articles, novels, etc.

In order to perform well at language modeling, a model must have strong syntactic and semantic representations of the language. For example, if asked to predict the next word in the phrase, "She jumped into the ... ", a human might pick the word "water", "lake", or "ocean", but would be less likely to pick "oven" and even less so to pick "smiling". This is because we understand that it is more syntactically appropriate to put a noun there than a verb, and that although "oven" is a noun, it doesn't fit semantically.

These features make language modeling a useful tool both for evaluating a particular model's ability to represent language internally, and as a means for producing good representations that can be used for other NLP tasks. [17] [35] [6] This idea of

pre-training a language model which can then be fine-tuned to a particular task is similar to techniques that have been popular in deep learning for image processing for some time. We can think of this as a kind of transfer learning, where language modeling produces general features and representations of language that can be transferred to a variety of tasks.

1.4 Hierarchical Language Modeling

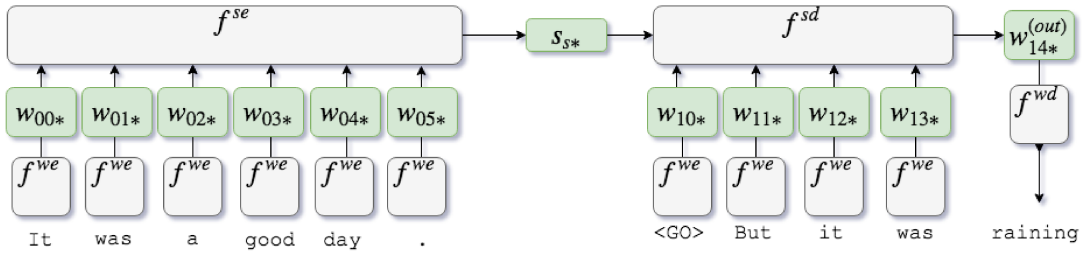

Figure 1.1: An overview of the hierarchical architecture. The word encoder, f^{we}, encodes each word to a vector for use by the sentence encoder, f^{se}, and the sentence decoder, f^{sd}. The sentence encoder produces a vector representation of the source sentence, and the sentence decoder produces a vector representation of the next word in the source sentence. The characters of the next word are then produced by the word decoder, f^{wd}.

A hierarchical language model is a generalization of a language model that contains an internal representation of the inherent hierarchical structures in natural language. These hierarchies include the composition of characters to produce words, words to produce sentences, sentences to produce paragraphs, and so on. A common approach is to explicitly model the relationship between characters and words, which has been found to improve performance over using character or word tokens on their own. [41] [20] [19] [5] [21]

The models in this work generally follow the architecture described in [41], but also add an extra level of hierarchy in order to represent sentences. As depicted in Figure 1.1, these models consist of 4 components: f^{we}, f^{se}, f^{sd}, and f^{wd}. The first, f^{we}, encodes a word by combining the characters that make it up. The input to f^{we} is an un-encoded word represented as a sequence of characters. It outputs the word encoded as a real valued vector, $\boldsymbol{w}_{sw*}$. The tensor $\boldsymbol{w}_{***}$ stores a number of sentences as

a collection of word-vectors, where the first dimension indexes sentences, the second indexes words, and the third holds the vector representation of that word. Now take $\boldsymbol{c}_{***}$ to be a tensor that stores a number of sentences such that each element identifies a single character. That is, the first dimension indexes sentences, the second indexes words, and the third indexes over characters within each word. We then represent the process handled by the word encoder as $\boldsymbol{w}_{sw*} = f^{we}(\boldsymbol{c}_{sw*})$.

Once each word in a sentence has been converted from a list of characters to a real-valued vector, each sentence can be passed to f^{se}, which encodes it by converting it from a list of word vectors to a single vector $\boldsymbol{s}_{s*}$. This process can therefore be represented as $\boldsymbol{s}_{s*} = f^{se}(\boldsymbol{w}_{s**})$. The third component f^{sd}, generates word vectors for the target sentence, whose index is $\boldsymbol{t} = \boldsymbol{s+1}$. It does so one word at a time, and at each time step takes as input the encoded representation of the source sentence, $\boldsymbol{s}_{s*}$, as well as encoded representations of all previous words in the target sentence. The encoded representations of previous words in the sentence are produced by f^{we}. Generating the word vector, $\boldsymbol{w}_{tw*}$, of the target sentence is hence formulated as:

$$\boldsymbol{w}_{tw*} = f^{sd}(\boldsymbol{s}_{s*}, \boldsymbol{w}_{t(w-1)*,\ldots,}\boldsymbol{w}_{t0*})$$

The final component, f^{wd}, generates the output characters that make up each word in the target sentence. It does so one character at a time, and for any given character takes as input the word vector generated by f^{sd} , as well as all previous characters in the word:

$$\boldsymbol{c}_{twc*} = f^{wd}(\boldsymbol{w}_{tw*}, \boldsymbol{c}_{tw(c-1)}\ldots, \boldsymbol{c}_{tw0})$$

The output of f^{wd} at a given time is a vector which models a probability distribution over all possible output characters. Hence each vector has a dimensionality equal to the number of possible characters, and each component of the vector represents the probability that a given character should be generated. In other words, $\boldsymbol{c}_{****}$ is a 4D tensor where the first dimension indexes over sentences, the second indexes words, the third indexes characters, and the fourth contains a probability distribution over all possible characters. Note that this tensor is separate from the 3D $\boldsymbol{c}_{***}$ mentioned earlier. The two are related in that $\boldsymbol{c}_{****}$ contains probability distributions over each character whereas $\boldsymbol{c}_{***}$ contains individual characters. Hence the scalar identifier for

an output character, $\boldsymbol{c}_{twc}$, can be obtained by sampling from the probability distribution represented by the vector $\boldsymbol{c}_{twc*}$.

Chapter 2

Model Architectures

2.1 Architecture overview

Language modeling experiments are performed with a hierarchical language model that explicitly models two levels of hierarchy inherent in natural language. The goal of these architectures is to encode a source sentence to a distributed vector representation, then decode that representation to produce a target sentence. In the case of language modeling, the target sentence is the next sentence in the text. That is, we encode a given sentence in a document and predict the one that follows. When paraphrasing, the target sentence is then one that retains the meaning of the source sentence but does not use identical wording and structure. These implementations explicitly model the concept that words are produced from characters, and sentences are produced from words. There are therefore four major components to the model: (1) the word encoder, (2) sentence encoder, (3) sentence decoder, (4) word decoder. The basic structure is displayed in Figure 1.1 The following sections describe each component in detail.

2.2 Character embeddings

The hierarchical architecture takes characters as input and produce characters as output. The character input representations are used by the word encoder, f^{we}, which produces word representations for the sentence encoder and decoder. They are also used as input to the word decoder f^{wd}, as representations of the previous characters in the word that is being decoded. The character output representation occurs as the final layer of f^{wd}, which is the output layer of the entire architecture.

For both input and output, the characters are represented by a vector whose dimension is equal to the total number of possible characters. Each dimension of these vectors corresponds to a specific character. This vector represents a one-hot

encoding, where the vector dimension corresponding to the present character is 1 and all other components are 0. This one-hot representation can then be matrix multiplied with a matrix of trainable parameters, $\boldsymbol{c}_{**}^{(emb)}$, to produce a real-valued vector, $\boldsymbol{c}_{c*}^{(emb)}$. This vector is a row selected from the matrix of parameters $\boldsymbol{c}_{**}^{(emb)}$, and is called a character embedding. For the output representation, the vector represents a probability distribution, where each component signifies the probability that its character should be produced. Multinomial sampling can then be applied to this distribution in order to produce characters at inference time.

2.3 Word encoder

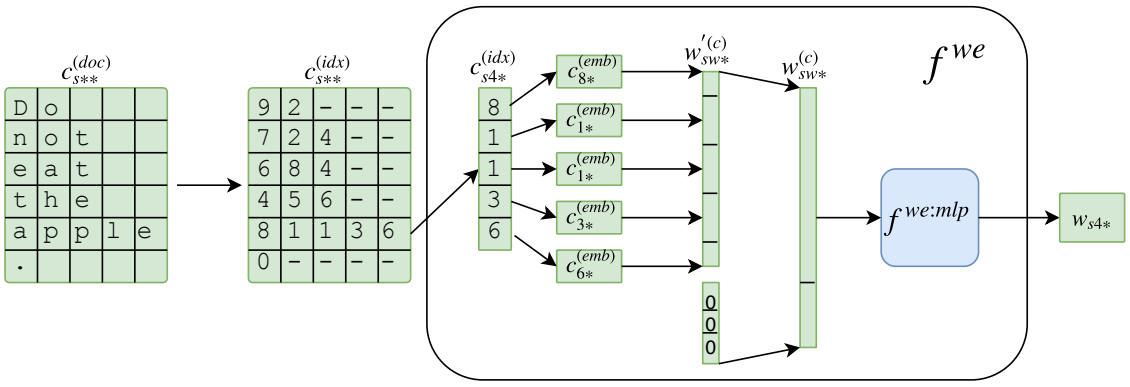

Figure 2.1: A graphical representation of the word encoder.

The task of the word encoder, f^{we}, is to take a variable length list of character embeddings, representing the spelling of a word, and produce a single vector to represent that word. Whereas popular methods use RNNs or CNNs for this task, we use a feed-forward MLP which has been shown to perform at least as well while requiring far fewer parameters. [41]

As shown in Figure 2.1, a word representation is composed from its characters. The matrix of trainable parameters $\boldsymbol{c}_{**}^{(emb)}$, contains a row for each possible character, where each row has the dimension $\boldsymbol{c}^{(dim)}$. A document is represented as a 3D tensor of characters denoted by $\boldsymbol{c}_{***}^{(doc)}$. This tensor is mapped to a new tensor called $\boldsymbol{c}_{***}^{(idx)}$, where each character has been mapped to a unique numerical index. This tensor, $\boldsymbol{c}_{***}^{(idx)}$, indexes sentences along its first dimension, words along the second dimension, and characters along the third dimension. The vector $\boldsymbol{c}_{sw*}^{(idx)}$ then represents the $\boldsymbol{w}^{th}$

word in the $\boldsymbol{s}^{th}$ sentence of the document, and is a list of character indices refering the characters that spell that word. These character indices are used to perform a lookup on $\boldsymbol{c}_{**}^{(emb)}$, extracting the rows that correspond to each character. These rows are considered the *character embeddings*.

Once the character embeddings have been selected, they are concatenated together in the order that they appear in the word. This results in a vector, $\boldsymbol{w}_{sw*}^{\prime(c)}$ with length $\boldsymbol{c}^{(dim)}\boldsymbol{w}^{(len)}$, where $\boldsymbol{c}^{(dim)}$ is the embedding dimension, and $\boldsymbol{w}^{(len)}$ is the word length in number of characters. This vector is then zero-padded or trimmed to a fixed length to produce $\boldsymbol{w}_{sw*}^{(c)}$. The fixed length vector, $\boldsymbol{w}_{sw*}^{(c)}$, is then passed through an MLP to produce the final word representation, $\boldsymbol{w}_{sw*}$.

2.4 Sentence encoder

Once the words of a sentence are converted to vector representations they are passed to the sentence encoder, f^{se}. The sentence encoder converts this sequence of vectors into a fixed length representation of the sentence. There are two phases to this process. The first reads each word vector sequentially and outputs a new vector which may incorporate information from previous words. The second phase combines these context aware word vectors with an attention mechanism to produce a compact representation for the sentence.

There are a variety of architectures that can be used for the first phase, such as Recurrent Neural Networks (RNN) , Gated Recurrent Units (GRU), Long Short Term Memory Cells (LSTM) , Convolutional Neural Networks (CNN), Transformers, etc. While the number of possible architectures for this task is limitless, this work compares two of the most popular sequence processing architectures, the GRU[8] and the Transformer[44], along with a novel approach, the *Self Attentive Recurrent Array* (SARAh), which uses far fewer parameters than a GRU and employs an attention mechanism similar to the Transformer, which allows it to keep a record of all previous states and refer to them as needed.

2.4.1 GRU

The GRU[8] is a popular RNN architecture that uses a gating mechanism similar the the LSTM, but uses fewer parameters. The purpose of the gating mechanism

is to control which state nodes are updated by the current input, and which nodes retain their state (memory) from previous timesteps. Given an input vector $\boldsymbol{x}_{t*}$ and a previous state vector $\boldsymbol{h}_{(t-1)*}$, the current state is computed as:

$$\boldsymbol{r}_{t*} = f^{(\sigma)}(\boldsymbol{x}_{t*}\boldsymbol{w}_{**}^{(rx)} + \boldsymbol{h}_{(t-1)*}\boldsymbol{w}_{**}^{(rh)})$$

$$\boldsymbol{u}_{t*} = f^{(\sigma)}(\boldsymbol{x}_{t*}\boldsymbol{w}_{**}^{(ux)} + \boldsymbol{h}_{(t-1)*}\boldsymbol{w}_{**}^{(uh)})$$

$$\boldsymbol{h}_{t*}^{(candidate)} = f^{(\phi)}(\boldsymbol{x}_{t*}\boldsymbol{w}_{**}^{(x)} + (\boldsymbol{r}_{t*} \odot \boldsymbol{h}_{(t-1)*})\boldsymbol{w}_{**}^{(h)})$$

$$\boldsymbol{h}_{ti} = \boldsymbol{u}_{ti}\boldsymbol{h}_{(t-1)i} + (1 - \boldsymbol{u}_{ti})\boldsymbol{h}_{ti}^{(candidate)}$$

Where $f^{(\sigma)}$ is the logistic sigmoid function, $f^{(\phi)}$ is the activation function of the GRU (popularly $tanh$), $\odot$ denotes element-wise multiplication, and $\boldsymbol{w}_{**}^{(rx)}$, $\boldsymbol{w}_{**}^{(rh)}$, $\boldsymbol{w}_{**}^{(ux)}$, $\boldsymbol{w}_{**}^{(uh)}$, $\boldsymbol{w}_{**}^{(x)}$, $\boldsymbol{w}_{**}^{(h)}$ are matrices of trainable parameters. The computed hidden state, $\boldsymbol{h}_{t*}$, is also the output of the GRU at that time step.

2.4.2 Transformer

The Transformer uses attention to combine each vector in the input sequence. As implemented in this work, the input vectors are presented as the rows of the matrix $\boldsymbol{x}_{**}$. A positional encoding vector, $\boldsymbol{p}_{i*}$, is added to each input. The positional encodings are implemented as position embeddings or, equivalently, as a linear feed-forward layer with one-hot inputs. The resulting position aware inputs are packed as the rows of $\boldsymbol{x}_{**}^{(pos)}$. Each is projected, using feed-forward layers, to a key, value, and query. Each resulting value is combined with the others in the sequence through an attention mechanism. For a given input, $\boldsymbol{x}_{i*}$, the output is computed as follows:

$$\boldsymbol{x}_{i*}^{(pos)} = \boldsymbol{x}_{i*} + \boldsymbol{p}_{i*}$$

$$\boldsymbol{q}_{i*} = \boldsymbol{x}_{i*}^{(pos)}\boldsymbol{w}_{**}^{(q)} + \boldsymbol{b}_{*}^{(q)}$$

$$\boldsymbol{k}_{i*} = \boldsymbol{x}_{i*}^{(pos)}\boldsymbol{w}_{**}^{(k)} + \boldsymbol{b}_{*}^{(k)}$$

$$\boldsymbol{v}_{i*} = \boldsymbol{x}_{i*}^{(pos)}\boldsymbol{w}_{**}^{(v)} + \boldsymbol{b}_{*}^{(v)}$$

$$\boldsymbol{y}_{i*} = f^{(attention)}(\boldsymbol{q}_{i*}, \boldsymbol{k}_{**}, \boldsymbol{v}_{**})$$

Where $\boldsymbol{y}_{i*}$ represents the output at position $\boldsymbol{i}$. Note that attention, which is described in section 2.4.4, is performed over the entire input sequence for each input token. The input tokens at the first layer are the word representations in $\boldsymbol{w}_{s**}$, and in subsequent layers the inputs are the ouput of the previous layer. The transformer therefore modifies the input sentence by allowing each of the word vectors to attend to any other in the sequence, producing a sequence of contextualized word representations.

2.4.3 SARAh

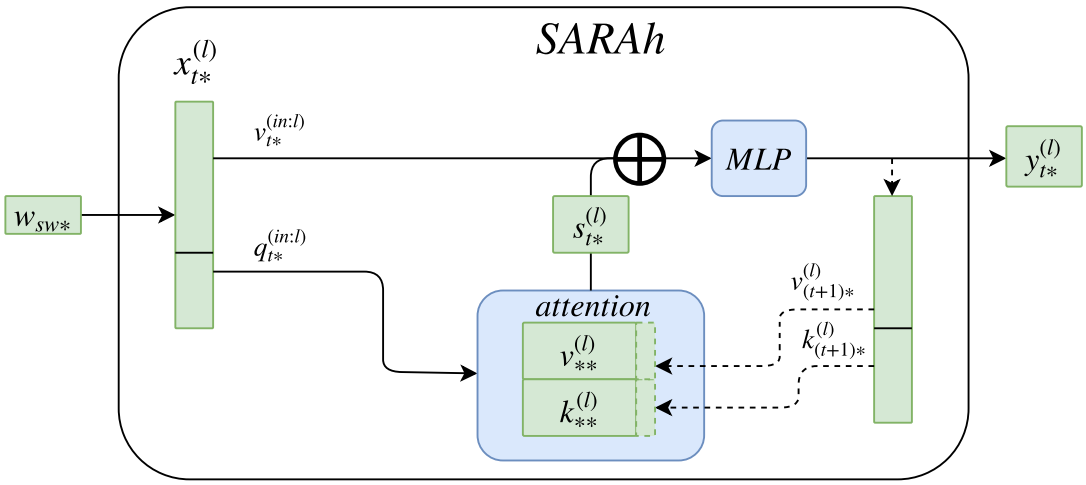

Figure 2.2: A graphical representation of a SARAh layer.

The motivation for the Self Attentive Recurrent Array is to allow the model to retain access to previous states. Whereas gated RNNs such as the GRU must decide at each timestep how much of the past information should be overwritten with the current input, the SARAh keeps track of all previous states. When a new input is provided, attention is performed over the history of states and is combined with the input representation. From there a new hidden state is produced and added to the array of past states, as well as passed on as output for the current time. This is in contrast to the Transformer, which performs a similar attention mechanism but over the entire input sequence. Whereas the Transformer must use positional encodings

on each of its inputs, the SARAh learns to encode position inherently and recursively, but can only attend to previous words. Given an input vector $\boldsymbol{x}_*$ and the history of previous states $\boldsymbol{h}_{**}$, the output is computed as:

$$\boldsymbol{q}_{t*}^{(in)}, \boldsymbol{v}_{t*}^{(in)} = f^{split}(\boldsymbol{x}_{t*})$$

$$\boldsymbol{k}_{**}, \boldsymbol{v}_{**} = f^{split}(\boldsymbol{h}_{**})$$

$$\boldsymbol{s}_{t*} = f^{attention}(\boldsymbol{q}_{t*}^{(in)}, \boldsymbol{k}_{**}, \boldsymbol{v}_{**})$$

$$\boldsymbol{v}_{*}^{(aware)} = \boldsymbol{v}_{t*}^{(in)} + \boldsymbol{s}_{t*}$$

$$\boldsymbol{y}_{t*} = f^{mlp}(\boldsymbol{v}_{*}^{(aware)})$$

$$\boldsymbol{h}_{**} \leftarrow f^{concat}(\boldsymbol{h}_{**}, \boldsymbol{y}_{t*})$$

At a given timestep, $\boldsymbol{t}$, the SARAh is passed an input vector. At the first layer this vector is $\boldsymbol{x}_{t*}^{(0)} = \boldsymbol{w}_{sw*}$, which is the output of the word encoder. Generally, the input to the $\boldsymbol{l}^{th}$ layer of the SARAh will be denoted as $\boldsymbol{x}_{*}^{(l)}$. This input vector is interpreted as though it were the result of concatenating a query vector to a value vector, and is separated accordingly. That is, the input vector is arbitrarily split into two separate vectors using f^{split}. The value and query sizes are $\boldsymbol{v}$ and $\boldsymbol{q}$ respectively. The incoming vector must have a dimension size of $\boldsymbol{d} \geq max(\boldsymbol{v}, \boldsymbol{q})$ so that it can be split into the respective parts. The value, $\boldsymbol{v}_{t*}^{(in:l)}$, is extracted from the first $\boldsymbol{v}$ elements of the input, whereas the query, $\boldsymbol{q}_{t*}^{(in:l)}$ is extracted from the last $\boldsymbol{q}$ elements. It is acceptable if the input size does not exactly match the sum of the value and query sizes, $\boldsymbol{v} + \boldsymbol{q}$. If the input size is smaller, the value and query simply overlap, if it is larger, the central elements of the input vector are ignored.

The SARAh also contains an array of all previous states in the form of the matrix $\boldsymbol{h}_{**}^{(l)}$. The state matrix has dimensions $(\boldsymbol{v} + \boldsymbol{q}, \boldsymbol{t})$, and holds a key-value pair for each timestep that has been processed so far. Each column of $\boldsymbol{h}_{**}^{(l)}$ contains a value of size $\boldsymbol{v}$ and a key of size $\boldsymbol{q}$. Formulated differently, each previous state contains a value and a key, and they are stored in the matrices $\boldsymbol{v}_{**}^{(l)}$ and $\boldsymbol{k}_{**}^{(l)}$ respectively. Again, these matrices have one entry for each previous timestep.

Using the query portion of the input vector, $\boldsymbol{q}_{t*}^{(in:l)}$, attention is then performed over the previous states, producing the vector $\boldsymbol{s}_{t*}^{(l)}$. The state summary $\boldsymbol{s}_{t*}^{(l)}$, is then added to the value portion of the input vector to produce a context-aware input

vector, $\boldsymbol{v}_{*}^{(aware:l)} = \boldsymbol{v}_{t*}^{(in:l)} + \boldsymbol{s}_{t*}^{(l)}$. This vector is then passed through a feed-forward layer which produces the output vector $\boldsymbol{y}_{t*}^{(l)}$ with size $\boldsymbol{v} + \boldsymbol{q}$. In addition to being passed as output, this vector is concatenated to $\boldsymbol{h}_{**}^{(l)}$ as the most recent state vector. This means that for successive layers of a SARAh, the keys of layer $\boldsymbol{l}$ double as the queries of layer $\boldsymbol{l} + 1$. This uses fewer parameters than computing them separately and was not found to impact performance.

2.4.4 Attention

Attention is applied internally over previous states within each SARAh layer, over the whole sequence within each Transformer layer, and to the outputs of the sentence encoder, regardless of the type of encoder that is used. In general, attention operates over three tensors: a query ($\boldsymbol{q}_{*}$), a set of keys ($\boldsymbol{k}_{**}$), and an associated set of values ($\boldsymbol{v}_{**}$). The number of keys is equal to the number of values ($\boldsymbol{k}_{**}$ and $\boldsymbol{v}_{**}$ have the same number of columns). The query is compared against each key via a dot product, producing a weight vector that that is used to attend to each value.

$$\boldsymbol{a}_{*} = f^{(softmax)}(\boldsymbol{q}_{*}\boldsymbol{k}_{*i})$$

$$\boldsymbol{v}_{*i}^{(weighted)} = \boldsymbol{a}_{i}\boldsymbol{v}_{*i}$$

$$\boldsymbol{v}_{i} = \sum_{j} \boldsymbol{v}_{j*}^{(weighted)}$$

The tensor $\boldsymbol{v}_{*}$ is therefore the result of applying attention to the all of the values contained in $\boldsymbol{v}_{**}$. In the case of internal attention used by SARAh and Transformer layers, the generation of the queries, keys, and values is described in their respective sections.

2.4.5 External Attention

Attention is also applied to the outputs of the sentence encoder. That is, the sentence encoder takes a list of word vectors in the form of the tensor $\boldsymbol{w}_{s**}$ and outputs a new list of word representations packed into the rows of the tensor $\boldsymbol{w}_{s**}^{(se)}$. The final task for the sentence encoder is to take this variable length representation of the input and convert it to a compact, fixed-length vector. This is accomplished by combining them through an attention mechanism. Each word vector, $\boldsymbol{w}_{sw*}^{(se)}$, is interpreted such

that it includes a key and a value. Therefore, each of these vectors can be split into the value $\boldsymbol{v}_{sw*}^{(se)}$ and the key $\boldsymbol{k}_{sw*}^{(se)}$. These are packed into the matrices $\boldsymbol{v}_{s**}^{(se)}$ and $\boldsymbol{k}_{s**}^{(se)}$.

In order to perform attention with these key/value pairs, a query vector must be produced and compared against each of the keys. We first take the mean of all the word vectors, computing the mean along the second dimension of $\boldsymbol{w}_{sw*}^{(se)}$ to produce $\boldsymbol{s}_{s*}^{(mean)}$. This representation of the input sentence is then passed as input to an MLP, the output of which is a query vector, $\boldsymbol{q}_{sw*}^{(out)}$. Attention can then be performed as described above.

When the sentence encoder contains multiple layers, this process is applied to the outputs of every layer. There is therefore one version of $\boldsymbol{s}_{s*}^{(se)}$ at each layer, which we will distinguish with the addition of the index $\boldsymbol{l}$ to the label: $\boldsymbol{s}_{s*}^{(se:l)}$. The sentence decoder, which has a similar architecture, is conditioned on the encoded sentence so that each layer of the decoder receives the encoded sentence from the corresponding layer of the encoder.

2.4.6 Multihead Attention

Multi-head attention is a applied to both the internal attention within each SARAh and Transformer layer, as well as to the external attention applied over the outputs of the sentence encoder. Multi-head attention is implemented similarly to [43], and allows the attention mechanism to be divided across multiple parts of a sequence. The number of heads is designated by $\boldsymbol{h}$, and the values, queries, and keys are each split into $\boldsymbol{h}$ separate parts. Attention is then applied separately to each of the $\boldsymbol{h}$ groups of values, keys and queries. The resulting vectors from each attention head are recombined through concatenation to produce a single vector.

2.5 Sentence decoder

The sentence decoder takes the first $\boldsymbol{n}$ words of a sentence and produces a vector representing the $(\boldsymbol{n+1})^{th}$ word. It does so using the same architecture as the sentence encoder, minus the encoder's external attention mechanism. The first $\boldsymbol{n}$ words are provided as vector representations after they've been processed by the word encoder, $f^{(we)}$. Each sentence is prepended by a special word that acts as the START token, while an END token is appended as the last word.

During training, each sentence to be decoded starts as a string. It is parsed into words, which are converted to vector representations using the word encoder. The sentence is then passed to the sentence decoder as the matrix $\boldsymbol{w}_{t**}$, where $\boldsymbol{t}$ is the index of the target sentence. For multi-layer configurations, we use the label $(de:l)$ to represent the $\boldsymbol{l}^{(th)}$ layer of the decoder. We denote the input at each layer to be $\boldsymbol{x}_{t**}^{(de:l)}$, and the output to be $\boldsymbol{y}_{t**}^{(de:l)}$. Note that $\boldsymbol{x}_{t**}^{(de:0)} = \boldsymbol{w}_{t**}$ and $\boldsymbol{x}_{t**}^{(de:l)} = \boldsymbol{y}_{t**}^{(de:l-1)}$. In order to condition the decoder on a source sentence, the encoded representation of that sentence is added to the inputs at each layer of the decoder:

$$\boldsymbol{x}_{tw*}^{(de:l)} \leftarrow \boldsymbol{x}_{tw*}^{(de:l)} + \boldsymbol{s}_{s*}^{(se:l)}$$

The output of the sentence decoder is a new set of word vectors: $\boldsymbol{w}_{t**}^{(de)}$. For each word vector that is input to the sentence decoder, it emits a word vector that is meant to represent the next word in the sentence. For example, when it is provided with the first word vector, $\boldsymbol{w}_{t0*}$, the emitted word vector $\boldsymbol{w}_{t0*}^{(de)}$ is taken to be a representation of the second word in the target sentence. The word vectors in $\boldsymbol{w}_{t**}^{(de)}$ are then passed on to the word decoder, $f^{(wd)}$.

2.6 Word decoder

The word decoder performs the inverse task of the word encoder, taking a vector representation of a word and producing a sequence of characters. The word vector, $\boldsymbol{w}_{tw*}^{(de)}$, is provided by the sentence decoder. Given this conditioning vector and a sequence of previous characters in the word, the word decoder outputs a vector representing the next character:

$$\boldsymbol{c}_{twc*} = f^{(wd)}(\boldsymbol{w}_{tw*}^{(de)}, \boldsymbol{c}_{tw(c-1)}, ...\boldsymbol{c}_{tw0})$$

Each emitted character vector, $\boldsymbol{c}_{twc*}$, represents a probability distribution over all possible characters, and can be sampled to produce a scalar character index, $\boldsymbol{c}_{twc}$. During inference, these character indices are passed back as input to the word decoder until it produces a STOP character. At that point the completed word is passed back to the sentence decoder, which produces the next word vector to be decoded. During training and evaluation, the character indices are provided by the datasets.

This work compares two implementations of the word decoder, the first is similar to the word encoder and uses only feed-forward layers. The second is a GRU. In both cases, the first $\boldsymbol{n}$ characters of the word are combined and represented by a fixed length partial-word vector, $\boldsymbol{w}_{tw*}^{(partial)}$. This partial word vector is then added element-wise to the corresponding word vector emitted by the sentence decoder: $\boldsymbol{w}_{tw*}^{(de)}$. The resulting vector is then passed through an MLP, the last layer of which uses a softmax activation to produce a probability distribution over all possible characters.

$$\boldsymbol{c}_{twc*} = f^{(wd:mlp)}(\boldsymbol{w}_{tw*}^{(de)} + \boldsymbol{w}_{tw*}^{(partial)})$$

The generation of $\boldsymbol{w}_{tw*}^{(partial)}$ differs depending on whether the MLP or the GRU implementation is used. They are described in section 2.6.1 and 2.6.2, respectively. Once it is generated and conditioned by the output of the sentence decoder, it is passed through an MLP wich outputs a probability distributiong for the next character.

2.6.1 MLP

The MLP based word decoder works similarly to the word encoder. It constructs $\boldsymbol{w}_{tw*}^{(partial)}$ by concatenating character embeddings together and clipping or zero-padding the result to a fixed length vector, then using this as input for a feed-forward layer. When predicting the $\boldsymbol{n}^{(th)}$ character of the $\boldsymbol{w}^{(th)}$ word, the indices of the previous characters are looked up in $\boldsymbol{c}_{sw*}^{(idx)}$. These indices are used to lookup the character embeddings in $\boldsymbol{c}_{**}^{(emb)}$ by indexing its rows. Therefore, extracting the embedding for the $\boldsymbol{c}^{(th)}$ character of the $\boldsymbol{w}^{(th)}$ word of the $\boldsymbol{t}^{(th)}$ sentence is formulated by:

$$\boldsymbol{e}_* = \boldsymbol{c}^{(emb)}_{(\boldsymbol{c}_{twc}^{(idx)})*}$$

These embeddings are concatenated together to produce $\boldsymbol{w}_{tw*}^{'(emb)}$. This vector will have a variable length due to the variable number of characters in each word. It is therefore either trimmed or zero-padded to a fixed length, producing $\boldsymbol{w}_{tw*}^{(emb)}$. This is vector is passed through a feed-forward layer to produce $\boldsymbol{w}_{tw*}^{(partial)}$.

$$\boldsymbol{w}_{tw*}^{'(emb)} = \boldsymbol{c}^{(emb)}_{(\boldsymbol{c}_{tw0}^{(idx)})*} || .. || \boldsymbol{c}^{(emb)}_{(\boldsymbol{c}_{tw(n-1)}^{(idx)})*}$$

$$\boldsymbol{w}_{tw*}^{(partial)} = f^{(wd:mlp-partial)}(\boldsymbol{w}_{tw*}^{(emb)})$$

2.6.2 GRU

The GRU is described in section 2.4.1, and selecting the character embeddings for the first $\boldsymbol{n}-1$ characters of the target words is performed in the same way as for the MLP version of the word decoder. However, rather than being concatenated together, the embeddings are passed in sequence to a GRU. This GRU outputs a vector for each character that is input. The output after the $(\boldsymbol{n}-1)^{(th)}$ character acts as $\boldsymbol{w}_{tw*}^{(partial)}$, which is then passed throught the MLP for predicting the $\boldsymbol{n}^{(th)}$ character.

Chapter 3

Language Modeling Experiments

3.1 Overview

The architectures described in chapter 3 (GRU, Transformer, SARA, MLP) are compared based on their language modeling performance. Each architecture compared here combines different versions of the sentence encoder, sentence decoder, and word decoder. The word encoder was kept constant because the feed forward architecture has been shown to perform well in previous work [41]. All models perform hierarchical language modeling over two levels of hierarchy (characters to words, and words to sentences). An encoder-decoder paradigm is used where the model's goal is to encode a source sentence and decode the target, which is the following sentence in a text corpus.

3.1.1 Data

The data used is a subset of the Gutenberg corpus [23], which is a collection of novels in the public domain. The dataset contains nearly 15 million sentences, which would take a considerable amount of time to train on available hardware. In order to allow time to make many comparisons of different architectures, about 10% of the data is randomly selected for training, and a further 1% for testing. This is done by randomly sampling the 3036 novels without replacement, resulting in 303 novels for training and 30 novels for testing. The total number of sentences is $1,458,297$ for training and $165,360$ for testing. This amounts to $25,453,338$ words for training and $2,820,012$ words for testing.

The data is parsed into sentences using the PunktSentenceTokenizer in the Natural Language Toolkit, which is a collection of NLP related tools available for the python programming language. [4] The maximum sentence length allowed is 30 words. To avoid having many truncated sentences in the training data, sentences longer than 30

words which contain one or more commas are split into sub-sentences at the comma(s). When there are multiple commas, the sentence is recursively split to maximize the length of the sub-sequence appearing before the split, up to 30 words. This greatly reduces the number of sentences that need to be truncated, and allows the models to train on long sentences by treating the second half of the sentence as the target sentence.

Target sentences shorter than 30 words are extended by appending up to 2 additional sentences, and trimming the sentence to 30 words. If a sentence is still shorter than 30 words, it is padded. Appending the following sentences to the targets in this way allows for fewer wasted computational cycles during batched training, in which all sentences in the batch must be padded to the length of the longest one, and provides more information to the sentence encoder during back-propagation.

Each sentence is then parsed into words using the following simple heuristic. Any consecutive string of alphabetical characters is considered a word, and any consecutive string of non-alphabetical characters is considered a word. Whitespace is removed, and words are trimmed to a maximum length of 20 characters.

3.1.2 Training

Training was performed on batches of 64 source/target example pairs. Examples were generated and placed into a queue during training. The document list was shuffled at the beginning of each epoch, and a buffer of 4096 examples was populated by interleaving examples from up to 128 documents at a time. This buffer was shuffled before sending examples to be queued for training. Filling batches with examples from multiple documents and shuffling them was found to be important for preventing overfitting on a particular style or vocabulary, especially at the end of an epoch. Training was performed for 15 epochs, at which point performance convergence was found to slow significantly.

Parameters are updated using gradient descent via backpropagation [36]. The gradients are computed to minimize an objective function, which in this case is the cross entropy ($f^{(xe)}$) between the final layer of the model, which represents a probability distribution over all possible characters, and the true character labels in the target sentence. The output layer of the model is last layer of word decoder, and

outputs the vector $\boldsymbol{c}_{twc*}^{(prediction)}$, which designates a propability distribution over all possible characters for the $\boldsymbol{c}^{th}$ character of the $\boldsymbol{w}^{th}$ word of the target sentence, which is the $\boldsymbol{t}^{th}$ sentence in the document. The labels are stored as one hot vectors in the corresponding tensor $\boldsymbol{c}_{twc*}^{(label)}$. The cost associated with predicting the target sentence is hence calculated as:

$$f^{(cost)}(\boldsymbol{c}_{t***}^{(label)}, \boldsymbol{c}_{t***}^{(prediction)}) = \sum_{w}\sum_{c} f^{(xe)}(\boldsymbol{c}_{twc*}^{(label)}, \boldsymbol{c}_{twc*}^{(prediction)})$$

$$f^{(xe)}(\boldsymbol{p}_*, \boldsymbol{q}_*) = -\sum_{i} \boldsymbol{p}_i log \boldsymbol{q}_i$$

There are a number of hyperparameters that are common to all models. The learning rate was kept fixed at 10^{-4}, and gradients were clipped to a maximum norm 5. The number of nodes used for character embeddings, as well as the size and number of layers in the word encoder is always the same. The number of layers in the sentence encoder and decoder is 2 unless otherwise stated. The size and number of MLP layers in the word decoder that follow $\boldsymbol{w}_{tw*}^{(partial)}$, which is the tensor representing the first $(n-1)$ characters of the word being decoded, is also kept constant.

3.2 Architecture Comparisons

A number of architectures are compared based on test set performance on the sampled Gutenber dataset. This work compares three main types of sentence encoder/decoder architectures, and two types of word decoder architectures. Different combinations of each are explored, as well as tweaks to hyper-parameters such as the number of attention heads used and the number of layers provided to the sentence encoder/decoder.

3.2.1 Word Encoder

The word encoder contains three feed forward layers after the character embeddings are concatenated. There are 306 character embeddings, each with 32 nodes and linear activations. The feed forward layers each have 512 nodes, with gaussian error linear units (GELU) [16], and a dropout rate of 10% [40]. The total number of parameters used by the word encoder is 863,296.

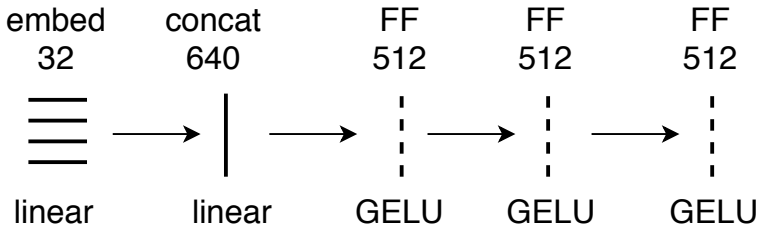

Figure 3.1: Details of each layer in the word encoder. The layer types and output dimension are indicated above each line (FF indicates a fully connected, feed forward layer). The activation function for each layer is indicated below. A dotted line indicates that the given layer has a dropout rate of 10%.

3.2.2 Sentence Encoder

Three main versions of the sentence encoder are compared, a GRU, a SARAh, and a transformer. Beyond the main architecture, the number of attention heads and the number of layers used is also modified. The number of layers used is two unless otherwise stated. Regardless of the main architecture of the sentence encoder, the output of each layer is passed as input to the next. In addition, once the sequence has been fully processed, the output of each layer is combined using an attention mechanism. This attention mechanism, described in section 2.4.4, uses a query of 256 nodes which is generated by taking the mean of that layer's outptus and passing it through a feed forward layer with GELU units. Each layer has a dedicated feed forward layer for producing the query, and the keys and values are split from the layer's oututs.

The GRU sentence encoder has 512 nodes per GRU layer with tanh activations and a dropout rate of 10% on the hidden state/output. Including the attention mechanism, each GRU layer uses 1,705,728 trainable parameters.

The SARAh sentence encoder has layers with 512 nodes dedicated to values and 256 nodes for the keys/queries. These layers have GeLU activations, and a dropout rate of 10%. The number of trainable parameters used by each SARAh layer is 590,866, which is less than half that used by the GRU layers.

The transformer layers have 512 nodes for a value, 256 nodes for keys, and a further 256 nodes for queries. These are produced by feed forward layers with GELU units and a dropout rate of 10%. Each layer also has position embeddings with linear nodes.

The first layer has 512 nodes per position embedding whereas the following layers have 768. Biases are also added to the position embeddings. The first transformer layer has 738,048 trainable parameters, and the following layers have 1,008,128. The first layer uses fewer parameters because it projects from the 512 node word encodings whereas subsequent layers project from the 768 node outputs of the previous layer.

3.2.3 Sentence Decoder

The architecture of the sentence decoder is always a clone of the encoder. The only difference is that it does not include the attention mechanism between layers. Instead, each layer is conditioned by the attended outputs of the corresponding encoder layer. This is accomplished by adding the attended outputs of the encoder layers to the inputs of the decoder layers. Note that attention is performed only once per source sequence, at the end of encoding. The decoder is not allowed access to individual tokens of the source sequence because the goal of the work is to train the encoder to produce a compact, fixed length representation of a source sentence.

3.2.4 Word Decoder

Two different word decoder architectures are compared, a GRU based architecture and an MLP based architecture. Both architectures produce a vector representation of the first $(n-1)$ characters of a word, add that representation to the corresponding output of the sentence decoder, then pass this conditioned representation through 4 feed forward layers, the last of which represents the probability distribution over possible predictions of the n^{th} character. The first 3 of these feed forward layers have GELU activations, a dropout rate of 10%, and sizes of 512, 512, and 256 respectively. The last layer has 306 nodes (one for each possible character), and a softmax activation function. The input character representations are 32 node character embeddings.

The GRU based word decoder combines characters with a 256 node GRU layer followed by a 512 node feed forward layer with GELU activations. The MLP based word decoder combines characters by concatenating the character embeddings together and passing the result through a 512 node feed forward layer with GELU activations. Not counting the last 4 feed forward layers common to both archictectures, the GRU based word decoder uses 363,328 trainable parameters and the MLP

based word decoder uses 337,472 trainable parameters.

3.2.5 Results

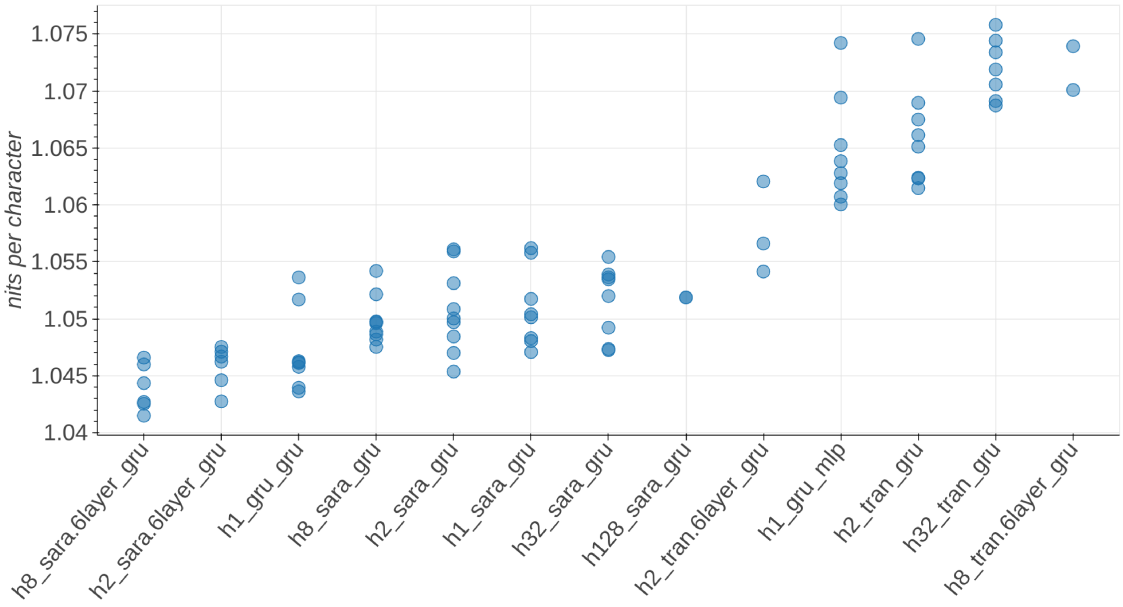

Figure 3.2: The test set performance for each architecture. Each dot represents the test set performance after training for 15 epochs on the given architecture. They are listed in order of best mean performance to worst, from left to right. The naming scheme is **h_s.d_w**, were h is the number of attention heads, s is the sentence encoder/decoder type, d is optionally the number of sentence encoder/decoder layers (defaults to 2) and w is the word decoder type.

Results from individual runs on a variety of architectures are shown in figure 3.2. Each dot represents the test set performance after one run of 15 training epochs. Performance is measured in nits per character, which is the unit of cross entropy when it is computed using the natural logarithm. The models are listed in order of the best mean performance to the worst, from left to right.

The MLP based word decoder does not perform as well as the GRU. The MLP based word decoder represents sub-sequences of the current word as a concatenation of the characters decoded so far. This may not be suitable, and it could be that the GRU is better able represent these subsequences. This is contrast to the case of word *encoding*, where the representation is only ever of whole words, and where an MLP has been shown to out perform a GRU when creating word representations. [41]

The Transformer does not perform as well as expected given previous research. [44][35] There are some notable differences between the implementations in those works as compared to this one. The first is that those models were not hierarchical and were not character based. Another difference is that whereas the implementations in this work were relatively shallow, previously published results were for much deeper models, and were trained on much larger datasets. It is also clear from figure 3.2 that the Transformer is sensitive to the number of attention heads.

The SARAh architecture performs similarly to the GRU, but uses only half as many parameters. It also does not look to be sensitive to the number of attention heads. Increasing the depth of the SARAh encoder and decoder from 2 layers to 6 layers increases performance slightly, while still using fewer parameters than a 2 layer GRU. Adjusting the number of attention heads does not result in much difference for the SARAh layers.

3.3 Stylistic Fine Tuning

3.3.1 Overview

It is beneficial to train language models on a large and diverse set of documents. However, there may be cases where it is desirable to have a language model that generates text with particular attributes, such as the characteristics of a particular author's style. Beyond author style there are more general distinctions, such as the differences between a story narrative, a description, an essay, a scientific paper, etc. This section proposes an approach for for producing language models with particular stylistic attributes, while still benefiting from training on large and diverse datasets.

One way to train a language model to produce text with a particular style is simply to train it exclusively on works with the desired style. However, there are often cases where there are few examples of works in the desired style, especially if that style is exhibited by a single author. Limiting training data to works in the desired style can therefore significantly limit the amount of training data available, which will in turn reduce model performance.

A better approach is to use a large and diverse dataset to pre-train a language model, then fine tune the model to the target style. This section presents a number

of experiments that show the value in doing so. It is difficult to objectively measure the degree to which a language model follows a particular style. To overcome this, we look at the model's performance on two test sets and look for two qualities which we interpret as successful stylistic fine tuning. The first test set contains a diverse set of styles, whereas the second contains only examples in the target style. The first desirable quality is that a pretrained model performs better on the target test set than a model which was not pretrained. This shows that some knowledge transfer has occured from the data that the model was pretrained on. The second desirable quality is that as performance on the target test set improves, performance on the diverse test set worsens. This shows that that the model is more likely to produce text resembles the specific target style than it is to produce more general styles.

3.3.2 Experiments

The model is pretrained on the reduced Gutenberg dataset described in 3.2. Fine tuning is performed on two different authors, which also appear in the Gutenberg dataset. The first is William Wymak Jacobs (WWJ), and was chosen at random. The works of WWJ are fairly representative of the rest of the Gutenberg dataset, making it difficult to assess what aspects of his style the model may be fine tuned to. The second author chosen for fine tuning is Charles Darwin. The works of Darwin were chosen because they are qualitatively different from the other, mostly fictional novels in the Gutenberg dataset. The largely descriptive nature and biological content of Darwin's work makes it more clear when a model has been fine tuned to that style of writing when it is compared to a model trained on the wider Gutenberg corpus.

The books attributed to WWJ that appear in the Gutenberg dataset are randomly divided into a training set of 86 books and test set of 11 books. This amounts to 91704 sentences for training and 3754 sentences for testing. The books attributed to Darwin were randomly divided into a training set of 17 books for training and 3 for testing, which amounts to 158, 261 training sentences and 40, 195 for testing. While fine tuning to WWJ, the model's performance is assessed against both the target test set and the larger, more diverse test set used in section 3.2. The more diverse test set is assessed only every 5 epochs in order to speed up the experiment, and is only assessed when fine tuning to WWJ. We also train a model on each of the

target datasets from scratch, without any pretraining. Both the pretrained model and the non-pretrained model have the same architecture, with 6 SARAh layers for the sentence encoder and decoder, two attention heads, and the same word encoder and GRU based word decoder as described in section 3.2.

3.3.3 Results

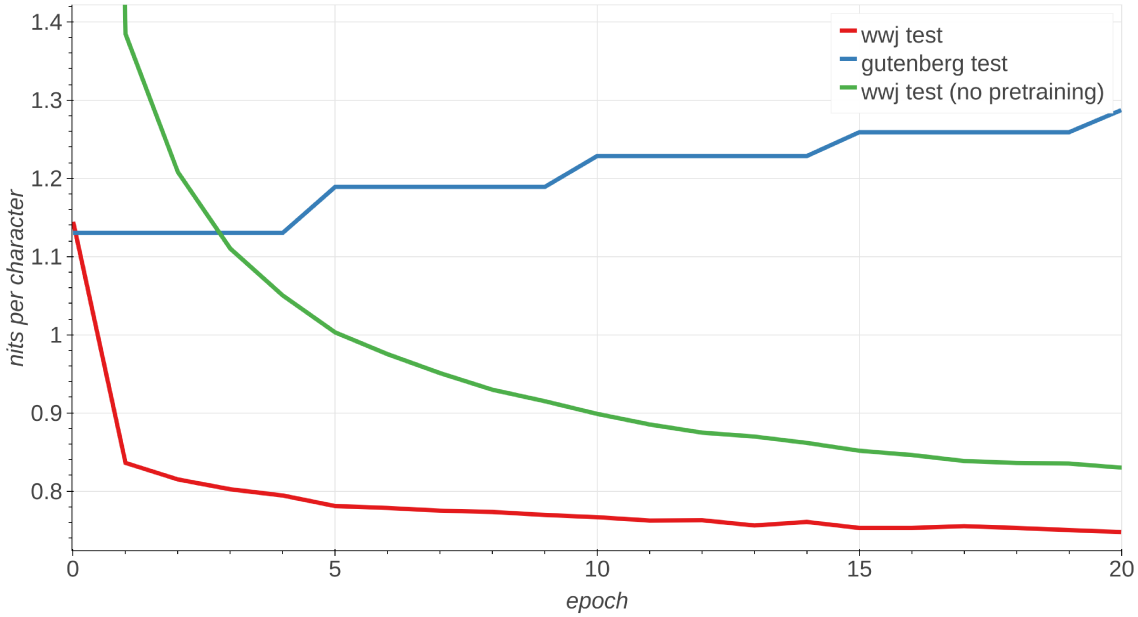

Figure 3.3: Test set performance during fine tuning on the works of WWJ, as well as the fine tuned model's perfomance on the Gutenberg dataset, and the performance of a model with no fine tuning on the WWJ test set.

Figure 3.3 shows WWJ test set performance after fine tuning a model that was originally trained on the reduced Gutenberg dataset. The WWJ test set performance of a model with no pretraining is also shown. The curves show that the pretrained model converges faster and performs better in the long run than a model with no pretraining. This suggests that there are aspects of the English language that were learned on the larger and more diverse dataset that the model was able to retain when fine tuning on the WWJ dataset.

Also shown in figure 3.3 is the pretrained model's performance on the Gutenberg test set. As fine tuning on the WWJ data progresses, performance on the WWJ test set improves while performance on the Gutenberg test set gets worse. This suggests

the model is learning to produce text that more closely resembles WWJ's style than that of other authors, which is the goal of stylistic fine tuning.

Figure 3.3 therefore suggests two important pieces of evidence for the effectiveness of fine tuning a pretrained language model to a particular linguistic style. The first is that the model transfers some knowledge learned during pretraining, performing better on the WWJ test set than a model that wasn't pretrained. The second is that performance on the Gutenberg test set declines, which suggests that the model is learning to generate text with attributes that are specific to WWJ, and which are not found in works written by other authors.

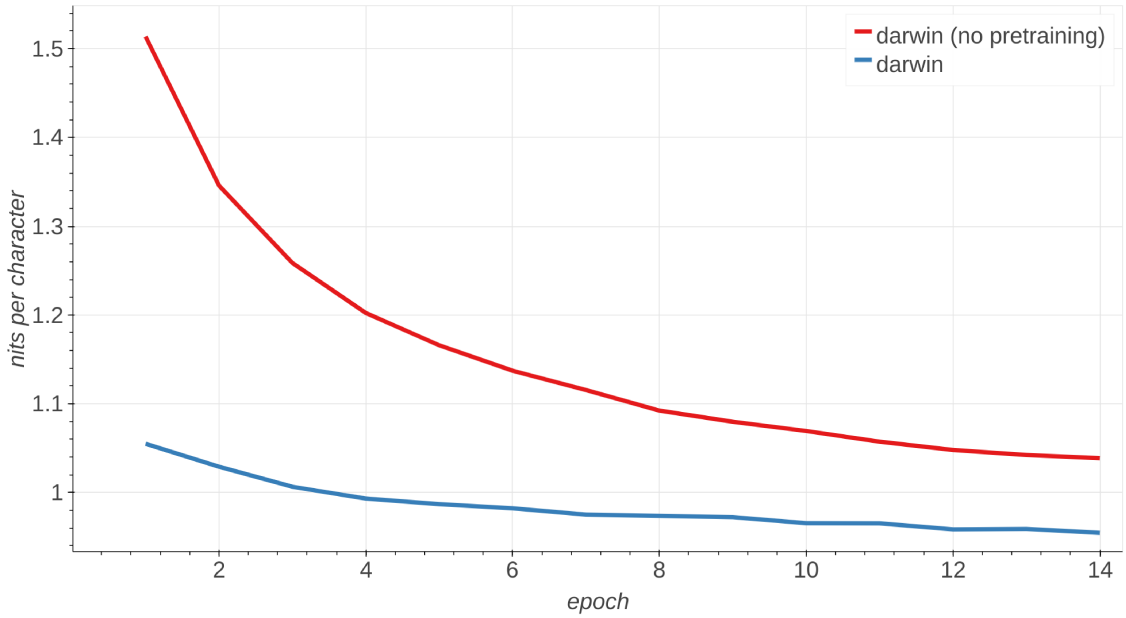

Figure 3.4: Test set performance during fine tuning on the Darwin dataset compared to test set performance of a model trained from scratch on the Darwin dataset.

Figure 3.4 shows test set performance during fine tuning on the Darwin dataset. Performance on the general Gutenberg test set was not assessed for this experiment in order to speed up the training process. Still, the figure shows that pretraining facilitates improved performance on the Darwin dataset, even though the style and content of those works is noticeably different from the narrative prose of the majority of novels in the pretraining dataset.

Table 3.1 contains example output from the model after pretraining and after fine tuning. Source sentences on which the output is conditioned were chosen at random

from the dataset, and are highlighted in blue. The output of the pretrained model, and each of its fine tuned variants, are displayed below the conditioning sentence.

Given that the output of the model is a probability distribution over the next character, two strategies are employed for generating a sentence. The first is to simply select the most likely character. Sentences generated with this strategy appear with a white background in the table. The second strategy employed is to sample from the probability distribution. Sentences generated by sampling are given a grey background in the table. When sampling, the softmax temperature on output layer's activations was kept at 1.0.

3.4 Discussion

This chapter provides many comparisons of architectures, as well as an investigation on the effects of fine tuning a pretrained model to a particular domain. Given that this requires training many models for multiple runs, doing so with large models and over very large datasets is not feasible with available hardware. For this reason, only a subset of the data is used, and the size of the models is restricted. These results are therefore not meant to compete with large models trained on huge datasets. The comparisons made between architectures in this chapter are to show that the SARAh performs at least as well as common architectures such as GRUs and Transformers. Futhermore, given the same state sizes and number of layers, the SARAh requires far fewer parameters. This allows us to increase the depth of the SARAh model while still using fewer parameters than a GRU.

We can clearly see the effect of fine tuning a language model to a given domain, such as a specific author, by looking at the language modeling performance. Fine tuning a model that was pretrained on a diverse dataset results in better performance than training a model from scratch on the target domain. While this suggests that some knowledge from the diverse dataset is relevant and retained, we see that performance on the diverse dataset decreases as fine tuning progresses. This is good, because it suggests that the model is more likely to generate text that resembles the domain it is being fine tuned to, and less likely to generate more generic text represented in the pretraining dataset.

By looking at the examples in table 3.1, we can see that the effects of fine tuning

are not trivial or merely measurable, but noticeable. For each conditioning sentence that was randomly selected, we see that the pretrained model and the two fine tuned models generate different outputs. The Darwin model tends to generate text with a more descriptive and academic tone. The Darwin model also generates content that references biology, such as anatomical or reproductive terms and descriptions of species. The WWJ model produces dialogue that more frequently contains the dialectic slang found his novels. For example: *'I ve ad enough o yourself"*. The table also compares the deterministic outputs of the model, generated by selecting the most likely characters, to the non-deterministic outputs generated by sampling the output layer. The deterministic outputs show that the differences between models in these example outputs is not merely chance. The sampled, non-deterministic outputs explore the models' generative space more freely, and we see that they are still able to retain the quality of the author they are fine tuned to.

Table 3.1: Examples of generated sentences. The source conditioning sentence is shaded blue. A white background indicates the most likely next character was selected during prediction, a grey background indicates the next character was sampled from the distribution produced by the model.

Condition	" I was at that time a sailor .
Pretrain	I had no doubt , and I had no one to be a good deal of it .
	You have been taken to our eyes fixed , of course .
Darwin	The sea is a most striking case .
	South Walder I should assume a best sight of the Potamogeians .
WWJ	I m not a bit of a soul ."
	I see Placed a day or two ago , and I used to have a knock in the joking ."
Condition	" And that ' s the fore hatch ?"
Pretrain	" I am a - going to see him , and I ' ll go to the stairs ,
	said Mr . Weller .
Darwin	– a single stripe of the skull , and a single sheep , and a few of the seeds of the same species ,
	Proc .
WWJ	" I ' m not going to ," said Mr . Kybird ,
	" I ' ve ' ad enough o ' yourself ," was the reply .
Condition	How did she know these things about this man , and why had she only now accidentally spoken of them ?
Pretrain	I had no doubt , and I had no doubt , and I had no one to be a good deal of her .
	To say , and her dislossing the situation and minding her !
Darwin	I was a most striking and a strong man , who was a most striking man ,
	As soon as I was till it had filled a bat ,
WWJ	I m sure I m not a bit of a soul ."
	Oh , she would join it !
Condition	A glance of artful intelligence came into Nott ' s eyes , which had remained blankly staring at Renshaw ' s apparently causeless hilarity .

Pretrain	’ I am sure I am sure ,’ said Mr . Weller , ’ I am a - having a - having a - hand ,
	I loved Mrs . Meg – but I was glad to be quick for a long pang ; and I filled the allusts ,
Darwin	I was assured by a few of the most ancient breeds of the same species ,
	(9/71.
WWJ	” I ’ m not going to be able to see you , my lad ,” he said ,
	” You can ’ t be George ,” said Tabitha , not to take a with her .
Condition	She was the daughter of the Italian Prince and Princess Monte Castello staying here .
Pretrain	I had a little strange , and I had a little of my own .
	Eleonorous , whom my sister - in - chief and her son of Sheer .
Darwin	I was assured by a few of the most ancient man , who , as I have seen ,
	(21/19.
WWJ	It was a strong , strong - looking man ,
	I m not busy .
Condition	but without losing her ideal romance in it .
Pretrain	I have no doubt , in the same way ,
	What answered , tortures , to their head , was no more known , so expressive ,
Darwin	I have seen a peach - bed , which had a single brown - brown ,
	Hence it must be true that , balled by a remarkable account of the just - represented dirt thus bearing in classification ,
WWJ	” I m not going to ,” said Mr . Vyner ,
	He had a fine spirit from the sharp - faced conclusions of the night .
Condition	There was something so queer in their story that our skipper took the law in his own hands ,
Pretrain	and the streets were so strong , and the streets of the streets ,
	again ; and the Vingue told me , and huck a deep flag , slowed his eyes .
Darwin	and the seed of the cock was a most strong one ; and the seed of the sea , which was a most striking ,
	in his work on ’ An Agar ’ (Marsupos existres), with five rather small hybrids ; they were purple ,
WWJ	an I saw the mate s eyes started as he saw the skipper s face , an the mate ,
	an I insisted under the way he had seen being making the most .
Condition	and looking in upon her sleeping children ; ” et ’ s a mercy and a blessing for them and – for – me .
Pretrain	I ’ m a - ha !
	You blank agensbass !
Darwin	I have seen a man who was a most ancient man ,
	” This 179 sort of periods , which I know of my peculiar exposed relation ,” and this is not far more intercrossing ,
WWJ	I ’ m a - standing in the ship ’ s bed , and I ’ m sure I ’ m going to ’ ave a ’ most .
	Only be longe , and we empty and all ; for before , the wharf ’ s got better .
Condition	A dash of rain against the window caused Rosey to lift her eyes from her book .
Pretrain	’ I am a - going to see her , and I ’ ll be a - having a - hand ,
	’ Upon every officer ,’ said Mr . Dick who has fallen in her eye , and sometimes , as soon as she was noisyless !
Darwin	I was assured by a few days ’ s face , and a few years since ,
	We thus see that some families furnished with liliae , called Evil - monkeyed ,
WWJ	” I m not going to ,” she said , at last .
	” A nice or widing fat chap knew what looks any part was before he could run after imself ,” she said ,
Condition	” Young man ,” said Nott , throwing a certain dignity into his habitual gesture of placing his hands on Renshaw ’ s shoulders ,

Pretrain	” I am a - going to see you , and I ’ ll tell you ,
	’ you see me good to some object .’
Darwin	” I have not seen a pig , and a single sheep , a man who was a man ,
	at middle south from the plant of China .
WWJ	” I ’ ll ’ ave ’ im ’ ere .”
	” he was old or natural –” ” I don t know now what you please would be ,” said the ” Maty ” ” as a matter of fact ,” said the girl ,
Condition	” I – think – I must leave to - night .
Pretrain	I am a good - humoured , and I am a - having a good - natured , and I am a - ha !
	I do it .
Darwin	I have seen a few of the most common fowls , and I have seen a peach - bird ,
	Many facts are of the same nature : neither in its acclimatisation than the useless .
WWJ	I ’ m not going to be able to see you , but I ’ m not going to be able to be a bit of a soul .
	I know I ’ m going to be a bit round and make it ’ ere for good .
Condition	” Co - la - te - ral is the word the big business sharps yer about call ’ em .
Pretrain	I ’ m a - havin ’ to see you , and I ’ ll go to the state o ’ the world ,
	I look at water , sir !”
Darwin	I have seen a peach - like breed of a breed of the same species ,
	(11/52.
WWJ	I ’ m going to see ’ im .”
	Who ’ ll have a knot of people ’ s hardships ?”
Condition	careless face .
Pretrain	’ I am a - going to be a - having a - having a - having a - having a - hand ,
	’ He has tended his advice to put up for him ,
Darwin	St .
	number during the brain in cavity .
WWJ	” I m not going to ,” ses the cook .
	He s a lold !
Condition	” Brother Silas speaks well ,” said Sister Parsons , with stridulous fluency .
Pretrain	” I am a - going to see him , and I ’ ll tell you ,
	” So his thing is a - go on fifty o ’ where he kill with the bar , with bright box , and a miraculous stop ,
Darwin	The same fact is the most important of all the cases of the same plant .
	Mr . H . Book on ’ Lesson ’ s ’ Bell - Canaries ’ 1865 page 375.
WWJ	” I m not a bit of a bit o mind .”
	” I wish we d do em something .
Condition	” Co - la - te - ral ,” he continued , emphasizing each syllable by tapping the fist of one hand in the open palm of the other .
Pretrain	” I am a - going to see you , sir ,” said I ,
	’ Sever my tea .’
Darwin	The same fact is a case of a single breed , and is a strong and strong case .
	In the Old Race with a parashierono .
WWJ	” I m a bit of a bit o mind , and I m not going to be able to see you .
	” There s no way of a words .”
Condition	With a start , he made a quick attempt to push aside her hands and rise .
Pretrain	’ I am sure I am sure I am , my dear ,’ said I ,
	’ My love !’

Darwin	I was assured by a few years , a few years ago , a few years ago ,
	Juns (Gould , a deswanting system).
WWJ	” I m not going to ,” she said , at last .
	He was all in a low , shock , angry , in face and group , where a small old seaman , with a lamb ,
Condition	Johnson , of Petaluny – him , ye know , ez had a game eye – fetched Flynn comin ’ outer meetin ’ one Sunday ,
Pretrain	and the story of the story , and the street ,
	or revolts , and the last , if he didn ’ t close ,
Darwin	and a single breed of sheep - tree ,
	its sheep With male Loudon ’ s smaller Ittens Dichuris ?”
WWJ	and a little bit of a strong , strong - looking man , who was a strange , strong - looking man ,
	and for a cancial moment he had to complain in the way two days .
Condition	and with a lofty simulation of examining the ceiling , said ,– ” Ahem !
Pretrain	I am a - going to be a - hand , and I ’ ll go to the state of a boy ,
	” Me , who is it ,”’ says he will not an open special little short temper ;– the gauntco ,
Darwin	a man who has a singular contrast , and who has seen a single species ,
	who were harder power ,” says Sir J . Luddrens .
WWJ	What s the matter ?”
	Nonsense !”
Condition	Nothing so far seemed inconsistent with her infelix reputation , but , strange to say , her other features were marked by delicacy and refinement ,
Pretrain	and the street was a little long , and the street ,
	Mr . Weller , ’ s whisky , and his face , the door - torpine ,
Darwin	and the fact of the seed - bird was a most striking one .
	that I saw no body ?
WWJ	and a strong , strong - looking man , who was a strange , strong - looking man ,
	and he was plying down the past of a lot as to who received a fierce man .
Condition	until even the very youthful scoffers in the last benches suddenly found their half - hysterical laughter turned to sobs .
Pretrain	I had no doubt that I had not been a little to be a strange ,
	It was even ,
Darwin	The same fact is a strong fact ,
	Hence I have seen the extraordinary rule of a mercue which was found in a lip by Ceramatish ,
WWJ	” I m not going to be able to see you ,” said the mate ,
	” He s all right , sir ,” said the mate , to Private Vyner .
Condition	Ferrers !
Pretrain	I am a - going to see you , and I ’ ll go to the state of the story .
	That ’ s I am !’
Darwin	The ” Beagle ” sailed from the sea .
	and is most quite so .
WWJ	I m not going to be able to see you again .”
	Not only ; he he he ges ” But at last , again , he eughterous the mate .
Condition	The stifling hold seemed again filled with struggling figures he had known ; the air thick with cries and blasphemies that he had forgotten .
Pretrain	’ I am a - going to be a - hair , and I ’ ll be a - having a - having a - hand ,
	In the most contemptuous effort that in the curvey of all this case ,
	The same fact was a most striking and a singular case .

Darwin	Certain aspect , as he had rejected ,
WWJ	He was a strong , strong - looking man , Altogether he was worn out on his way to a tall , short - up dog who would have gone to a sea - glass .
Condition	The next moment he was gone , and Mr . Sleight again became absorbed in his papers .
Pretrain	' I am sure I am sure ,' said Mr . Pickwick , ' I am a - having a - having a good - humoured , ' You will be a little , my dear , sir !'
Darwin	I was assured by a few of the most ancient and striking facts in the seeds of the seeds of the same species , (11/35.
WWJ	" I m not going to be able to see you ," he said , slowly . Two men lay all on a companion , but realised resolution .
Condition	Or , ye might do this !"
Pretrain	" I am a - going to see you , my dear , and I am a - having a - having a good deal , Miss Ben shook her tone .
Darwin	I have seen a few breeds of the same species , and the seeds of the same species , We may now seeing , in a temperate sense ,
WWJ	" I m not going to ," said the other , Flower shook his head .
Condition	A slight color came into his cheek .
Pretrain	' I am sure I am sure I am ,' said I . I was not the same at the door .
Darwin	The sea was a little less than two years old , and the seed was a little less than two - third of the seed - like shell . When I resembled some duty , he joaded with his whole young and if woitturing ,
WWJ	" I m not going to ," he said , slowly . " Call you ard !"
Condition	A quick flash of color over her brow and cheek - bone , as if a hot iron had passed there , and a certain astringent coyness ,
Pretrain	and the street - stairs , and the streets of the streets , I believe I was , with a secret spiny - quattled in my life ,
Darwin	and the feet were a little left . with good hair on her skull , sometimes an imperfection , and a true tree suggested to him .
WWJ	she sat in the state - room . and the return of the hatches , caused by that of them , were now walked up the quay and went slowly forward with the news .
Condition	" Brother Silas speaks well ," said Sister Parsons , with stridulous fluency .
Pretrain	" I am a - going to see him , and I ' ll tell you , " But one who is very happy ."
Darwin	The same fact is the most important of all the cases of the same plant . The plants are much more far more striking on the horse than others .
WWJ	" I m not a bit of a bit o mind ." " It s a fraBpled ed and a subject , an he s got a good sea - like boy !"
Condition	and all the instinctive tact which was born of his gentle thoughtfulness for others , to repress a shrinking .
Pretrain	I had no doubt , and I had no doubt , He told the subject to the powers of the shell ,
	I have seen a few of the most ancient breeds of the same species ,

Darwin	” With these same mystery , it was failed to change ; and any other must have been the case ,
WWJ	He was a fine - looking man ,
	A few men shook a little ,
Condition	The building of the barn had been arrested when the half load of timber contributed by Sugar Mill brethren was exhausted ,
Pretrain	and the streets were so strong , that the streets were so strong ,
	and was stabbed by my prize - books that short writing .
Darwin	and the seed - like striped striped with a single brown or sheep - like shell ,
	as he found a touch in a leaf ; nor did it , as I was able to show from Mr . William Leogra ,
WWJ	and the skipper was a strong , strong - looking man ,
	and a half - bag of housekeepons he ran in and fell across the door on a taste behind .
Condition	” And you ?”
Pretrain	” I am a good man , and I am a man , and I am a man ,
	” Now I was haughtily all right ; I felt it .
Darwin	– a single stripe of sheep .
	– salt living at an extreme reception .
WWJ	said the other , in a low voice .
	repeated Mrs . Church .
Condition	had this independent communication with the alley .
Pretrain	’ I am sure , my dear ,’ said Mr . W .
	You see , in the revived troops of gold ,
Darwin	I have seen a few of the most common forms , which have been seen in the same part of the sea ,
	One of these meaght I descended together an action of pressing - inheritance , each of which cost the South Africans the nops ,
WWJ	” I m not going to be able to see you , my lad ,” said the mate ,
	The mud was led by the noise of the mate .
Condition	He paused a moment , and then , as a new idea seemed to be painfully borne in his round eyes ,
Pretrain	and the street was a - going to be a - born .
	the need of mortal agents in her voice , and those of her dear Lord Traddles .
Darwin	he was a most striking one ; and he was a most striking one .
	” is not made out with a crimson sark entire in their chief part .”
WWJ	sat down and sat down to see the skipper .
	turned a few words to his ladies and gazed in a choking fashion at the boat .
Condition	that diverged from the straight practical path before them .
Pretrain	I had no doubt that I had not been a little to be a proper state of the same state of the state of the state of the state of the state ,
	It is of no use with great pleasurations for the cruel terms ; others ,
Darwin	The same fact is a strong fact ,
	I do not know how far a plan was the presence of a bear ; and as the cells of the silk - mottled extension or party bear fine - pollen ,
WWJ	The skipper , who had been a soul of the sea , was a source of a strong , strong - looking man ,
	A little couple of past ten minutes later they shook her head , and , slowly finished the cook .
Condition	of which a few untouched bales still remained against the wall .
Pretrain	’ I am sure , my dear ,’ said Mr . W .
	Make a ” Neares ,” ’ louved out of nine men ’ s propence to the ” gift ., and a small , when these trades .
Darwin	The seeds of the seeds of the same species , as well as of the same plant ,
	Secondly , no one looks at a shorter regular earlier than 1500, as had BRACES been of any kind .

WWJ	” I m not going to be able to see you , my lad ,” said the mate ,
	Mrs . Bunker said , ” I said , she s a jeweller ,” and followed the Metcalfe .
Condition	” I was there an hour or so – him and me alone – discussin ’ trade .
Pretrain	I have no doubt he was a good deal of his own .
	For I have been a worse than one this , in her set the Port - grouped ; and I don ’ t .
Darwin	I was a most striking and a strong and an interesting case .
	Dr . Quacker , who was return to this or I would treat a man , got him to see how the history of the ear .
WWJ	I ’ m not a bit of a bit of a soul .”
	I suppose that ’ s talking to him of it ?”
Condition	’ Thou ’ st larned to love another thou ’ st broken every vow we ’ ve parted from each other and my bozom ’ s lonely now oh is it well to sever such hearts as ourn for ever kin I forget thee never farewell farewell farewell .’
Pretrain	’ I am a - going to be a - having a - having a - having a - hand ,
	(Mr . Morton kindly told a still to Dover , as Hollistre could not then .
Darwin	I have seen a few years ago , and I have seen a few years ago ,
	Knife ’ s great need be known : like our blue pigs , a finger fertilisation have been come from any before 24 1/2 years .
WWJ	” I saw the captain ’ s eyes and the startled start of the skipper .
	” I had to get away from Bill ,
Condition	and what he had deemed her innocent craving for information concerning it , a consequence ?
Pretrain	’ I am sure , my dear ,’ said Mr . W .
	’ I saw him say and sat again .’
Darwin	I was assured by a man who had a single short - faced shoot ,
	A singular ass of a male , a remarkable son , rose , and a man , was a reversion to Nonybear ,
WWJ	” I m not going to ,” she said , at last .
	” Else he s riding shouting at me ,” she muttered , catching her hair .
Condition	” He ’ ll last as long as he has need of you ,” he said respectfully .
Pretrain	” I am a - going to see him , and I ’ ll go to the same .
	” I am afraid you doen ’ t know that he ’ d have a - having to be a boy , not very like a child ’ s .”
Darwin	He who will not say when he has seen a single species ,
	The individuals of the Cochin ’ s roots are generally purple , and it has almost probably been erumented by the paruline of the eyes .
WWJ	” I ’ m not a bit of a bit of a bit o ’ things .
	The captain sceamed into the direction of Mr . Silk ’ s purple - top ,
Condition	was the grand barkeeper of the hotel .
Pretrain	’ I am a - going to see you , my dear , my dear ,’ said Mr . Micawber ,
	Mrs . Hammer softly at the father ’ s , firm ,
Darwin	The sea was a little less than two - thirds of the sea ,
	On a gland about thirty miles in an enlarged row ,
WWJ	” I m not going to ,” he said ,
	” I don t think I could make a circumstance on you , my galley ,” said Miss Hartley ,
Condition	” Ef you ’ re thinkin ’ o ’ skippin ’ afore I ’ ve done with yer ,” said Nott with labored gentleness ,
Pretrain	” I ’ m a - goin ’ to see you ,
	’ I may dinner up to my hand , to be a mere shill .
Darwin	” I have not seen a peach - bird , and a single sheep , a single sheep , a single sheep ,
	for it is the apple ’ s ’ wing - hotter ’ s ?

WWJ	” and I ’ ll ’ ave a bit o ’ the way ’ ere .”
	” and all she ’ s soft - spirited , and a best place on it .”
Condition	to meet the requirements of the goods and passengers who were once disembarked on what was the muddy beach of the infant city .
Pretrain	I had no doubt that I had not been a little to be a great deal of the same state of the state of the state of the state of the state ,
	For a time , at a small part of the stair ,
Darwin	I have seen a few of the most complex cases of the same kind ,
	Owing to this fact ,
WWJ	The skipper , who had been a soul of a strong , strong - faced man ,
	Her ship s glee , which there was a straw at the Sannett s Blue Puff ,
Condition	he remained for a moment mute , trembling and irresolute .
Pretrain	’ I am sure I am sure I am , my dear ,’ said I am a - having a - having a good - humoured ,
	’ You ?’
Darwin	The seed of the sea , which was a man , was a most strong and so - called tree .
	The tree , when compared with an organ , smuded its gaucher , and soon constanted him , as she was not on the colour in the cow .
WWJ	” I m not going to be able to see you , my lad ,” he said ,
	” Well , I m not going to go ,” he said .
Condition	and the bulk of their household goods apparently in the process of removal .
Pretrain	I have no doubt that the same thing is a proper one ,
	’ But Mr . Nicaaro is conscious of my try or in my young companions .’
Darwin	The same fact is the most important of all the cases of the seeds of the same species ,
	One year the successive rocks were not subjected to foreign breeds and one has been so influented with some large scenery , and ,
WWJ	” I m not going to be able to see you , my lad ,” said the mate ,
	” It s a lot o day - times to give me a click , sir ,” said a thoughtful boy ,
Condition	as it were , still lovingly hand in hand , from the earth forever .
Pretrain	’ I am a - going to be a - having a - hand ,
	’ There was Mr . Malay ’ s pipe on the French ,
Darwin	The seeds of the seeds of the same species , as well as the seeds of the same species ,
	It is not , however ,
WWJ	” I m not going to be able to see you , my lad ,” said the mate ,
	” What do you want ?”
Condition	He drew the dressing - gown around him with an air .
Pretrain	’ I am sure I am , my dear ,’ said I ,
	’ You may walk into his wheel on it – as I think of Coming , am I ?’
Darwin	The seed - trees , as I hear from Mr . W . W . Brown ,
	The amirtesions were power : a young man was an infant to all the one blood , when he went on bitter ,
WWJ	” I m not going to ,” he said , slowly .
	” Didn t I get my art to you ?”
Condition	The footsteps approached nearer , and a tall figure suddenly stood in the doorway .
Pretrain	’ I am a - going to be a - hand , and I ’ ll be a - having a - hand ,
	I dwitningly lay on the ear , and uttered a little roof of the sheet of men in my head ,
	The sea was a little less than two miles in length , and the sea was so strong that it was a most common .

Darwin	It is confluent , but not a mode of affection , whenever one of these , with object , had to the same spot much .
WWJ	The skipper , who was a man of some state - pocket , was a source of anxiety ,
	It was clear that the universal old lady , who was in a slow arms , was eyeing her in hot course about what he meant .
Condition	" They waz counterfeits got up by them Dutch supercargo sharps for dealin ' with the Injins and cannibals and South Sea heathens ez bows down to wood and stone .
Pretrain	I ' m a - havin ' the best to see the same ,
	(Dead ' s three black things came at the lines of my past .
Darwin	The same fact is a strong case of a single breed ,
	It is surprising that a gooseberry goes on blepted .
WWJ	I ' m a ' most as bad as a man ,
	Somebody ."
Condition	III Although Gideon Deane contrived to find a nest for his blanket in the mouldy straw of the unfinished barn loft , he could not sleep .
Pretrain	' I am a - going to see you , my dear ,' said Mr . Micawber ,
	He was not at all sufficient to think of the recale between all this changeman and the rest ,
Darwin	I have seen a peach - bed , which had been seen in a state of nature ,
	Absolutely the seedling , and so gastrian from a resisted in breeding - body , appears at first several differences .
WWJ	He was a fine - looking man ,
	Twenty years ago , the noise of the chest of change over returning work , he appeared to be a dawn ,
Condition	He stopped and hesitated ; he might have said more , but the widow ,
Pretrain	and the stranger was a little of a little better than a strange and so much .
	' I shall come out at rest on the escape ,' the severe , I do know letters for the son ,
Darwin	and he was a most striking one ; and he was a most striking one ,
	for we extended to another kind : on the same subject the young various deaf - duck ,
WWJ	who had been a few days of the way ,
	who was sitting at the door to the steamer , said that there was no concern .
Condition	separated from their nurses in the sudden onset of the storm , had taken refuge there .
Pretrain	I had no doubt that the streets were all the same ,
	' In the beaver , as I recognize , pure and too , there was a hope to look at once ,
Darwin	The same fact , as I hear from Mr . W . W . Brown ,
	In three other period of Prof .
WWJ	The skipper , who had been a soul of a strong , strong - looking man ,
	" Why don t it go last ?"
Condition	" I was sayin ' that Mr . Ferrieres didn ' t happen in while the young feller was there – eh ?"
Pretrain	" I ' m a - having a - having a - having a - has a - having a - has a - having a - has a - has a - has ,
	' Now , that I have to put a sight of my Son , sir ,' I asked !
Darwin	I was assured by a few of the most ancient breeds of the same tribe ,
	I do not know the fact that both Dahliaea , which was so pleasant , in several seasons five plainly were perished in the whole mouse .
WWJ	" I ' m not going to ," said Mr . Kybird ,
	" It ' s got nothing to do with you , mate ," said the captain .
Condition	At the expected sight they unconsciously stopped – unconsciously disappointed .
Pretrain	' I am sure , my dear ,' said Mr . Weller , ' I am a - having a - having a - having a good - humoured ,
	' Iw ' r - toe fer that ' Thing we don ' t raise me !'
	The sea was a little less than two - thirds of the sea ,

Darwin	We chiefly treated one of its standard of force , ” by ’ s with mere great distribution .”[quoted on selection .
WWJ	” I ’ m not going to ,” said Mr . Kybird ,
	It is a posy evening , an I ll ask you to tell Hoggs always that I told you ,
Condition	and one or two by a door cut through the ship ’ s side communicating with an alley on either side .
Pretrain	I have no doubt that the state of the state of the state is a little of the same state of the state of the state of the state of the state ,
	’ It ’ s the first , very wreck , and comes with it in a man , some sweep his grey .’
Darwin	The seeds of the seeds of the same species , as well as the seeds of the same species ,
	All the results yielded by the roots of the white - crowning goose on small and small - flowers ,
WWJ	” I ’ m a - coming to - night ,” said Mr . Kybird ,
	Miss Smith , who was in a cordeur to her , was now married to her struggles , and , after a long pause ,

Chapter 4

Paraphrasing

4.1 Overview

Most work involving deep learning and paraphrases has focused on classification, detecting whether or not two sentences have the same meaning. [18][7][1][45][11][15] Much of this work relies on datasets of known paraphrases, such as the Microsoft Research Paraphrase Corpus,[10], the Twitter paraphrasing corpus[25], or the MSCOCO dataset[29]. While detecting paraphrases is challenging problem, and one with many applications, the problem of generating paraphrases is less commonly tackled. One reason for this is the lack of very large datasets of paraphrases on which to train. This is in contrast to the similar task of translation, for which massive datasets have been compiled. Both translation and paraphrasing can be formulated as a sequence to sequence task, where a source sentence is consumed by a model and represented in such a way that a target sentence can be generated with the same meaning, but with different words.

Whereas some work has been done on generating paraphrases from supervised training sets, [34] [27] [49] [14] this chapter outlines some preliminary work to establish the value in using unsupervised techniques to train a paraphrase generator. A number of works have already shown promise at unsupervised translation, [2][24][50] and recent work has shown that pretrained language models provide representations of text that are useful for quickly attaining state of the art results on a variety of NLP tasks. [35] [9]

Many of these recent advancements rely on a key idea in deep learning: neural networks optimize their objective function by learning to transform their inputs to a new latent space in which the relevant features are more easily expressed. For NLP applications in particular, neural networks are able to take a very sparse input representation in the form of as text tokens such as words, and map them to an internal representation in which the spatial relationships between these tokens represents their

relative meanings.

Encouraging a network to learn an internal representation that captures the semantic space of its inputs requires training it on an objective function for which that semantic space is valuable. And there is a deceptively simple task that has this quality: modeling the probabilities of occurences words, given some input of nearby words. This is inspired by the concept, often attributed to Wittgenstein [46][47], that the meaning of a word is encoded in its usage. Hence by training a neural network to learn the usage of a word by predicting nearby words, the neural network learns to encode meaning and other lexical features in a latent representational space. This concept applies not only to word representations [3][31][32][21], but also to phrase and sentence representations in relation to language modeling [19][22][44], translation [8][48][30], and many other language related tasks.

In light of this, this chapter seeks to use language modeling with deep neural networks to produce latent space representations of sentences that can be used to generate paraphrases. An architecture similar to the one described in previous chapters is used to hierarchically encode sentences from the character level, through words, and finally to sentences. This is accomplished by training the network to predict a sentence from the previous one in a large text corpus. Once the encoder produces strong sentence representations, its weights are frozen. At this point the decoder is re-trained to decode the input sentence from the latent representation produced by the encoder. Two techniques are used to encourage the decoder not to reproduce the input sentence exactly, but instead to produce paraphrase. These are adding noise to the encoded sentence representation, as well adjusting the softmax temperature when sampling the output layer.

4.2 Pretraining

4.2.1 Overview

In order to pretrain a model to produce good internal sentence representations, they are trained to perform hierarchical language modeling on a large dataset. Language modeling in this section is implemented similarly to what is described chapter 3.

Table 4.1: Example of the word decoder's targets when generated for a single word vs for a fixed length window of 8 characters.

word index	single word targets	fixed length window
0	It	It was a
1	was	was a go
2	a	a good d
3	good	good day
4	day	day .
5	.	.

Three models, slightly modified from that chapter, are compared. The first is relatively shallow but wide, the second is narrower but much deeper, and the third is both wide and deep and is trained with noise added to the sentence encoding. They are trained on the Toronto Book Corpus [52]. The models are trained to predict the next 3 sentences, up to a maximum of 30 words, from a source sentence.

A modification to the word decoder's targets is also made. Rather than predict only the characters of a single word, the word decoder is made to predict the next 20 characters in the sentence, regardless of how many words that covers. As a consequence, the characters that follow those of the current word overlap with the next word prediction. The motivation for this is to encourage the sentence decoder to produce context vectors that capture the semantic space of the word to be decoded, not just the characters of that word.

The sentence decoder produces a vector for each word in the target sentence, which we represent by $\boldsymbol{w}_{tw*}^{(out)}$. This vector is then decoded into the characters of that word by the word decoder. If the word decoder must only produce the characters of that word, then the latent space encoded by $\boldsymbol{w}_{tw*}^{(out)}$ may be encouraged to simply represent those characters. However if the word decoder must decode $\boldsymbol{w}_{tw*}^{(out)}$ to produce a number of words, then that vector may be encouraged to represent the meaning of the word rather than just its character composition. Table 4.1 contains an example of how these targets look.

4.2.2 Data

The models are trained on the Toronto Book Corpus. [52] This is a large collection of novels containing over 70,000,000 sentences. The sentences in this dataset are already

parsed such that one sentence appears per line, and all words have been lower cased and delimited by whitespace. For this work, sentences over 30 words are then trimmed and/or split using the same algorithm described in section 3.1.1.

4.2.3 Model Details

Three configurations of the model are trained, a wide model, a deep model, and a third which is both wide and deep. The wide model has four 2048 node feed forward layers in the word encoder, 4 SARAh layers each with 2048 value nodes and 512 key/query nodes for the sentence encoder and decoder, and a 512 node GRU followed by five 2048 node feed forward layers, plus the output layer, for the word decoder. The deep model has eight 512 node feed forward layers followed by a single 1024 node feed forward layer for the word encoder, 10 SARAh layers each with 1024 value nodes and 256 key/query nodes for the sentence encoder and decoder, and a word decoder with a 512 node GRU followed by two 1024 node feed forward layers, followed by eight 512 node feed forward layers plus the output layer. All feed forward layers, including those that make up the SARAh layers, have GELU activations and a dropout rate of 10%.

The third model results from a series of modifications made to the wide model, which is then retrained in order to fine tune the new layers. Two 2048 feed forward layers are added to the word encoder. Whereas in previous models, each sentence encoder layer combines its outputs with an attention mechanism and passes this on to the sentence decoder, this model is modified by first passing the attended outputs through a unique projection layer of 2048 GELU nodes with no dropout. The outputs of the last layer of the sentence decoder, are passed through an extra 2048 GELU node feed forward layer with 10% dropout. Finally, the word decoder is given 4 new feed forward layers before the output layer: a 1024 node layer followed by three 512 node layers, each with GELU nodes and 10% dropout. Finally, the sentence representation is perturbed during training by sampling from a normal distribution as is described in section 4.5. The scaling factor on this distribution's variance is set to 1. This model is referred to in the rest of this work as the Wide, Deep and Noisy (WDN) model.

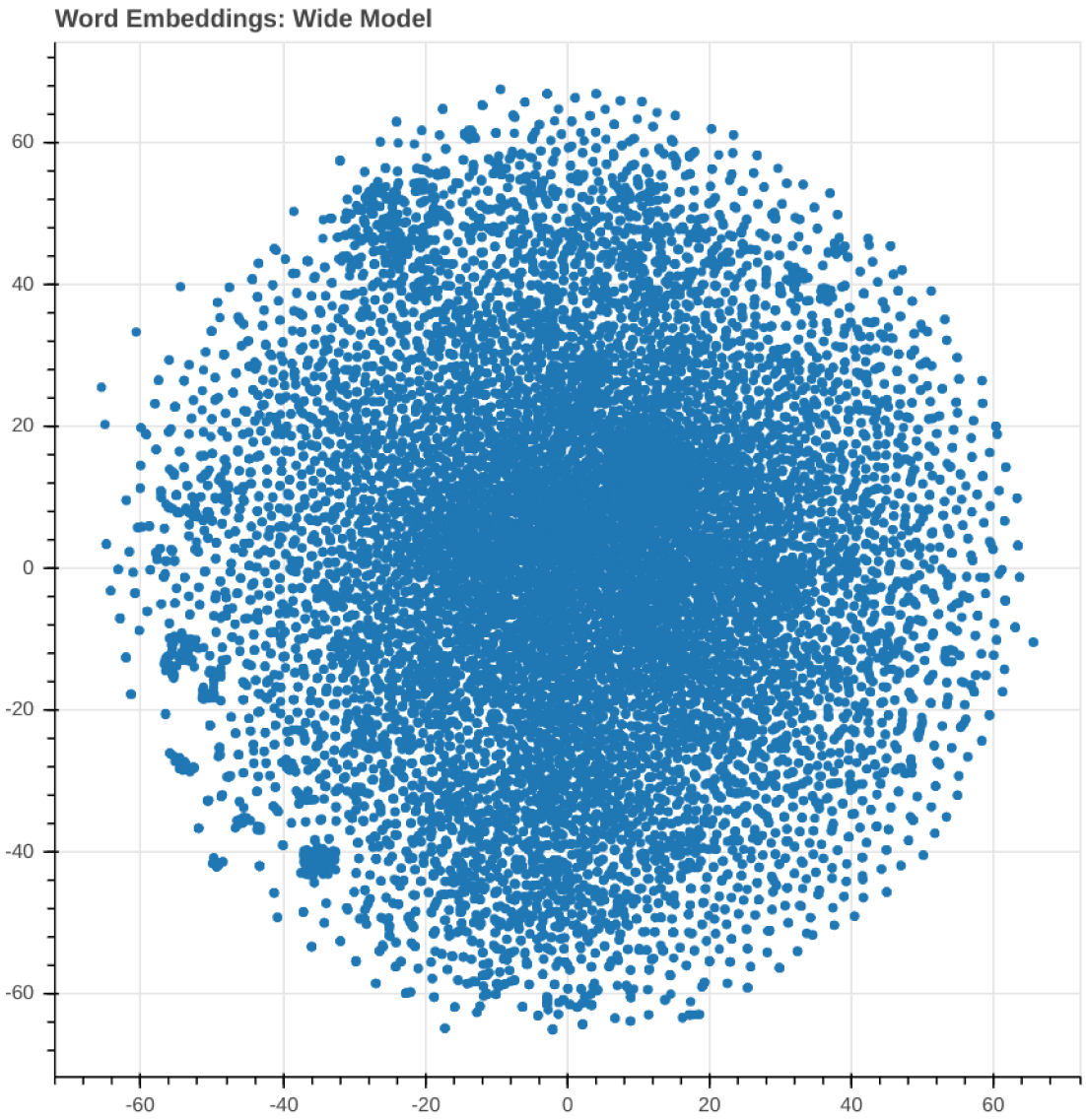

Figure 4.1: The 20,000 most common words in the Toronto Books dataset, embedded by the wide model and reduced to two dimensions with tSNE.

4.2.4 Word Representations

The importance of model depth for character based NLP models is evident when inspecting the word level representations of the wide versus the deep model. The word representations emitted by the word encoder can be inspected by applying t-SNE [42] to them in order to project them from the high dimensional (1024 or 2048 dimensions) down to 2 dimensions. This process is applied to the 20,000 most common words in the dataset. Figures 4.1 and 4.2 show the embedding space for these dimensionality

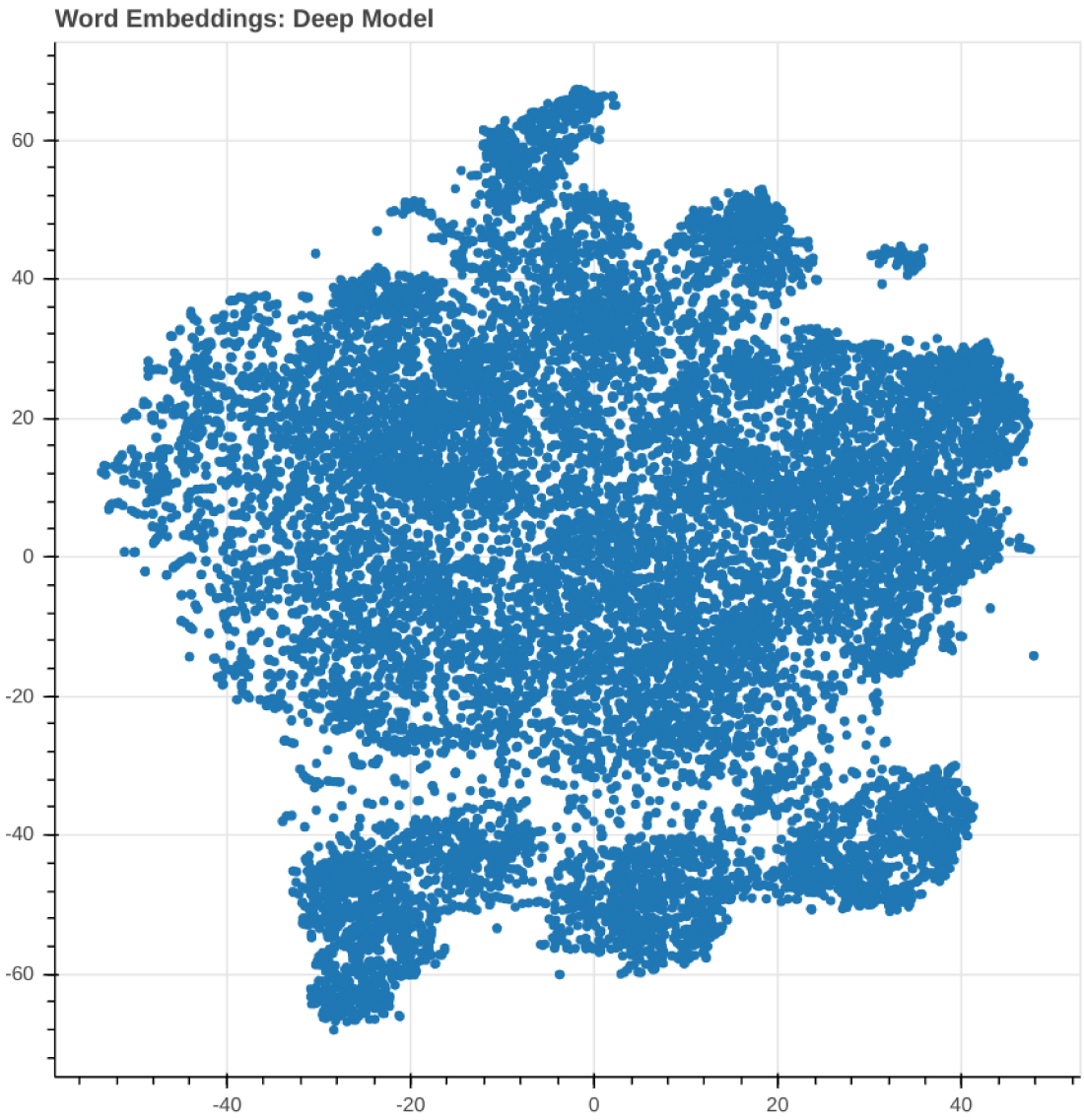

Figure 4.2: The 20,000 most common words in the Toronto Books dataset, embedded by the deep model and reduced to two dimensions with tSNE.

reduced word representations for the wide model and the deep model, respectively. There is very little structure in the embedding space for the wide model. They are distributed fairly evenly in a circular region, with a higher density in near the centre. The word representations for the wide deep have much more structure, with distinct regions corresponding to syntactic classes.

The differences between the wide and deep models' ability to encode words from characters goes beyond the global structure of the embedding space. Figures 4.3 and

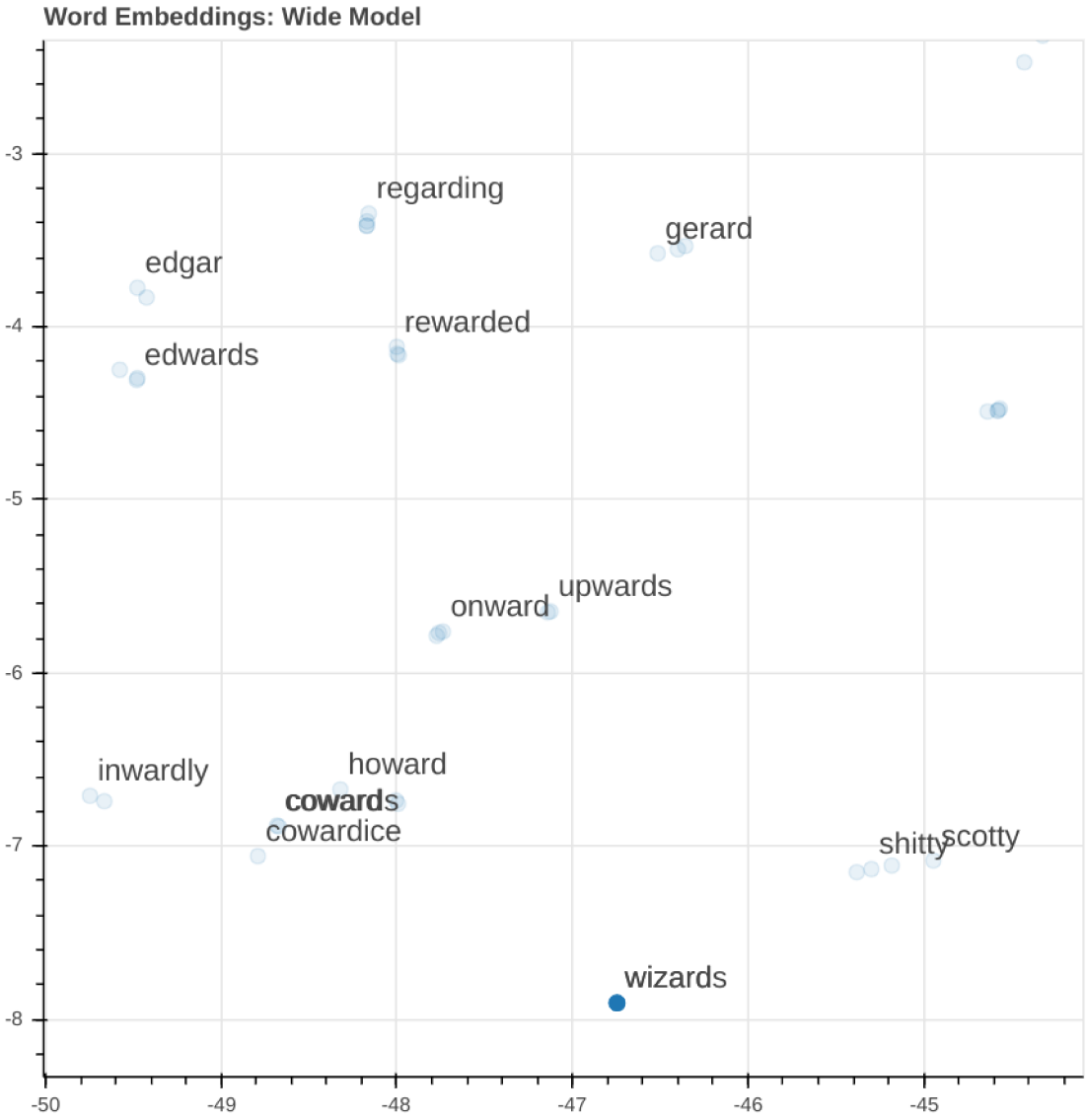

Figure 4.3: Showing the local word embedding space near the word 'wizards' for the wide model. The nearby words have little to no semantic relationship to 'wizards'. Instead, most of the words in region contain the string 'ard' starting at the 4th character position.

4.4 show the local embedding space near the word 'wizard' for the wide and deep models respectively. The wide model's local embedding space contains words that have similar characters in similar positions. For example, words that appear near the word 'wizard' in the wide model's embedding space tend to contain the string 'ard' starting at the fourth character. The deep model's embedding space represents more semantic relationships to nearby words, and we can find 'wizard' near related nouns

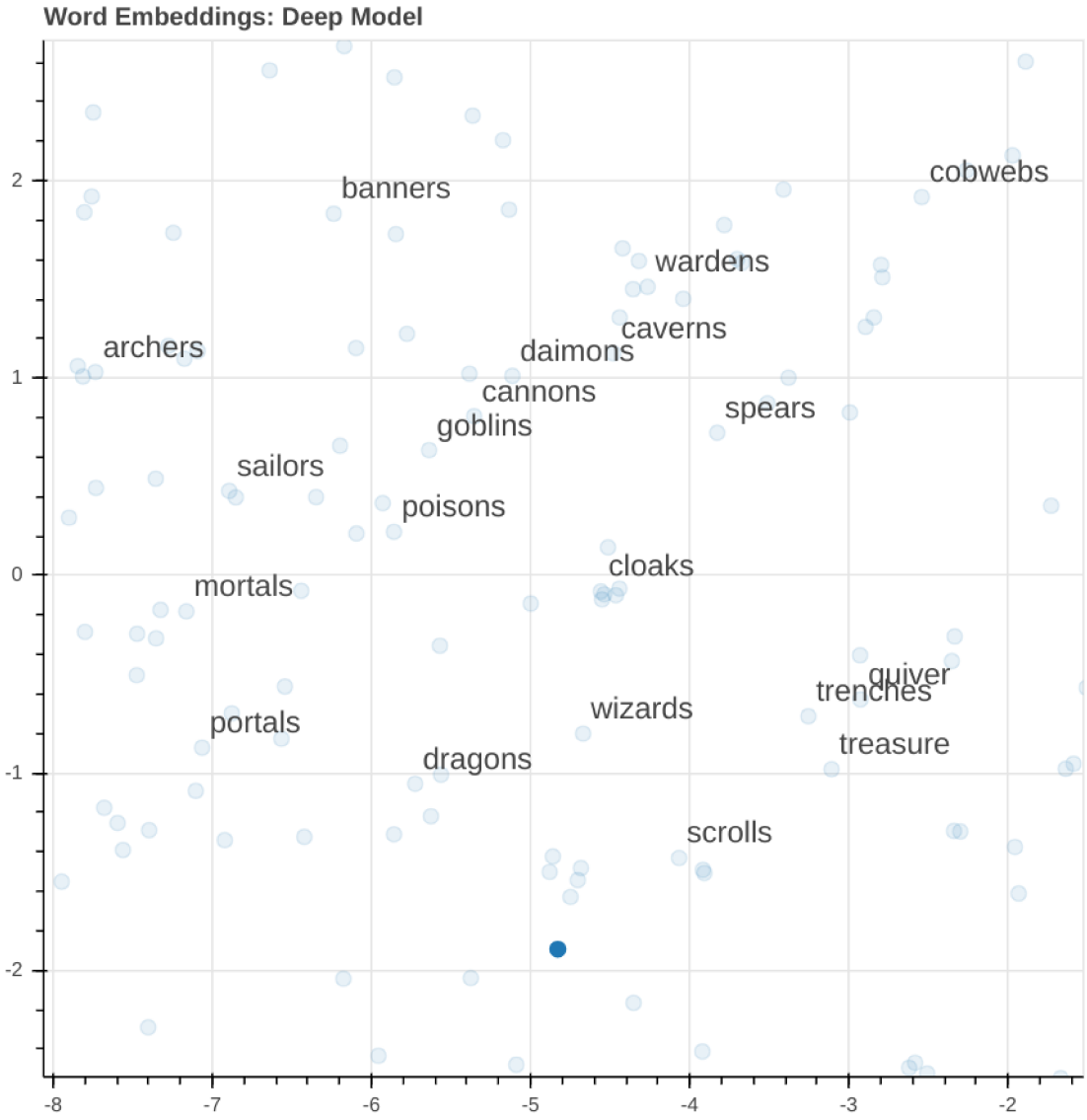

Figure 4.4: Showing the local word embedding space near the word 'wizards' for the deep model. Nearby words are semantically related in that they are likely to be found in similar contexts, such as in fantasy novels.

such as 'dragons', 'scrolls', 'cloaks', and 'goblins'.

That the wide model positions together words that are spelled similarly is unsurprising. The base representation of the word encoder is a vector of character embeddings concatenated together in the order of the word's spelling. That representation clearly puts words with similar spellings near each other, so the wide model is simply failing to transform that representation to a more useful semantic space. The wide model has only a few feed forward layers, which are no enough to learn the

necessary mapping. The deep model, on the other hand, has 9 feed forward layers through which it can transform the character based representation to one that better expresses semantic relationships. Furthermore, the deep model is able to learn these representations with far fewer parameters and in many less training steps. The deep model's word encoder has only 2.6M parameters while the wide model's word encoder has 9.8M parameters. Also, these t-SNE projections were made after 1.5M training steps for the word model and after only 0.2M steps for the deep model.

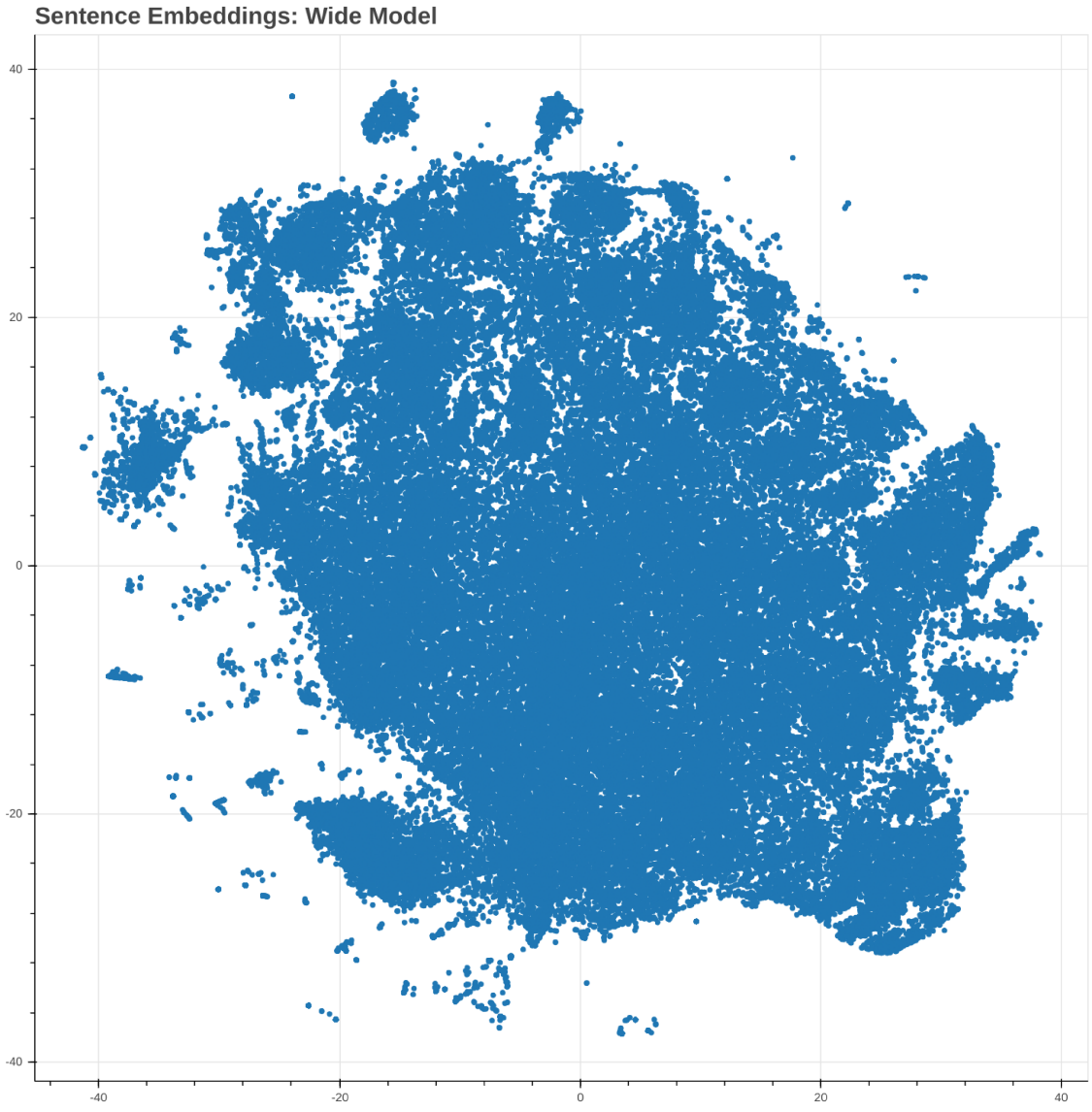

Figure 4.5: Showing the general structure of the sentence embeddings space for the wide model.

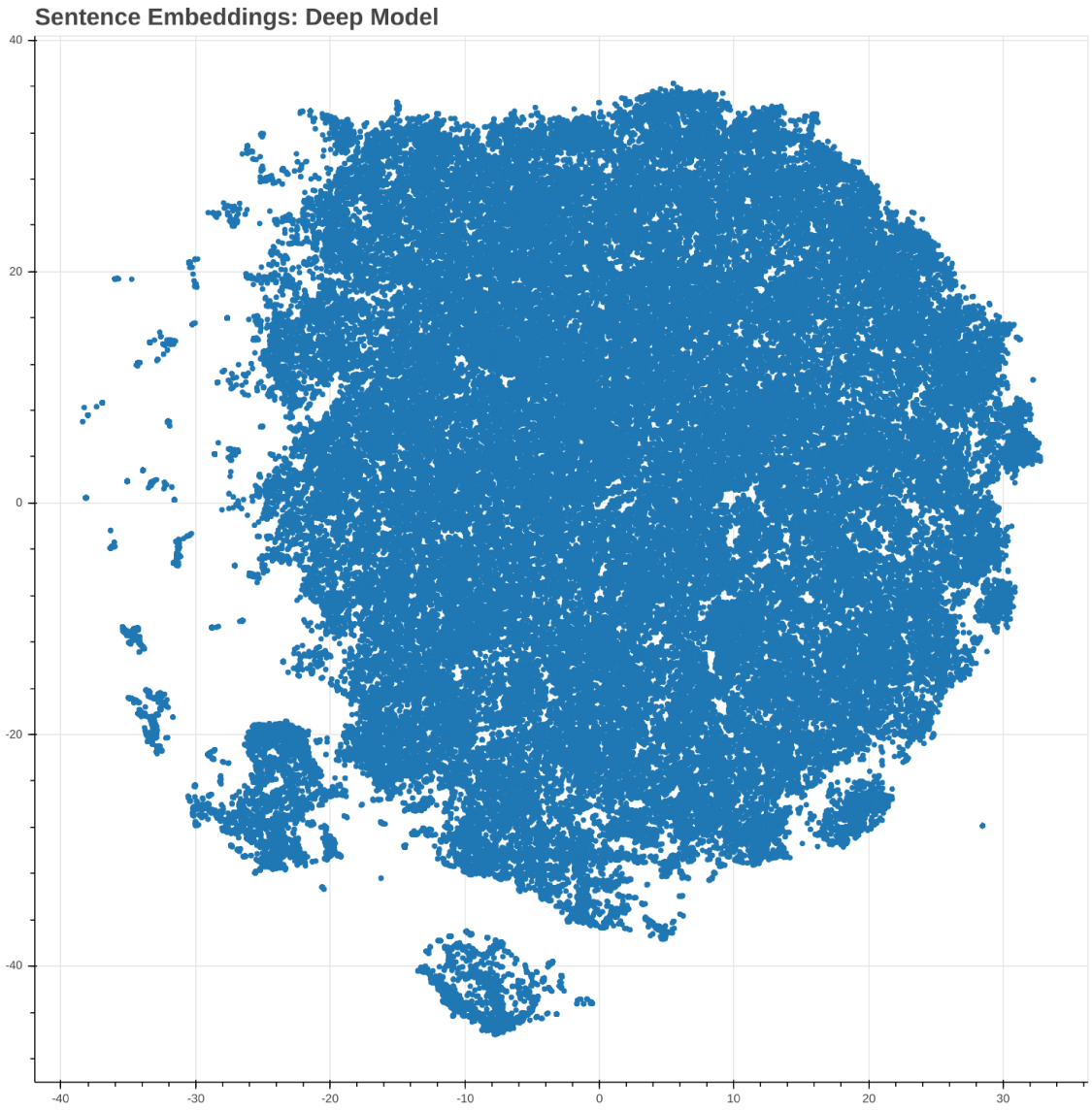

Figure 4.6: Showing the general structure of the sentence embeddings space for the deep model.

4.2.5 Sentence Representation

The sentence representations are not as straight forward to evaluate, because they are split over multiple layers of the sentence encoder. Each layer of the sentence encoder performs attention over its own outputs and passes the result along to the corresponding layer in the sentence decoder. Therefore each layer has an opportunity to attend to different aspects of the input sequence. As an approximation of the sentence representation, we look at the attended outputs of the final layer of the

sentence encoder.

Figure 4.5 and figure 4.6 show t-SNE projections of the sentence encoding space for the wide and deep models, for 100,000 randomly selected sentences. The wide model has slightly more defined clusters than the deep model, which is contrary to the word embedding spaces. Regardless, the global structure of the sentence embedding space for both models reflects the syntactic properties of the sentences. For example, short pieces of dialogue are clustered together, as are sentences that represent a question, a description of a person/object/scene, an exclamation, etc.

Figure 4.7 and figure 4.8 show the local embedding space near the sentence *"the rain was a little heavier and the wind had become noisy"*, for both models. Both models group this sentence with other descriptions of weather, as well as with descriptions of lighting and sound. The deep model contains, interspersed in this region, a number of descriptions of vocalisations of pain, which relates to nearby descriptions of other sounds. The wide model does not group such sounds together in this region, but it does include multiple descriptions of people's eyes.

It should be noted that both models group together sentences that are semantically related, but which do not simply contain similar key words such as 'rain'. For example, the wide model encodes the following two sentences in a nearby space: *"The midday sun had long passed and the dark clouds above were filled with rolling thunder."* and *"he snow had stopped at long last but the sky had not cleared yet."*. These sentences both describe a sky which, after some event, are left in overcast. However, they don't share any key words with each other. One describes that the the *"sky had not cleared yet"*, whereas the other describes *"dark clouds above"*. This suggests that, although the wide model's word encoder does not produce semantic representations at the word level, the sentence encoder is able to compensate and generate semantic representations at the sentence level.

Finally, the sentence representations for the wide, deep, and noisy (WDN) model show better semantic clustering than either the deep model or the wide model. Figure 4.9 shows the global structure of the same 100,000 randomly chosen sentences after they have been encoded and t-SNE projected down to two dimensions. This shows much more clearly defined clusters than the encoding space for the other two models. Inspecting the local embedding space also shows an improved ability to encode

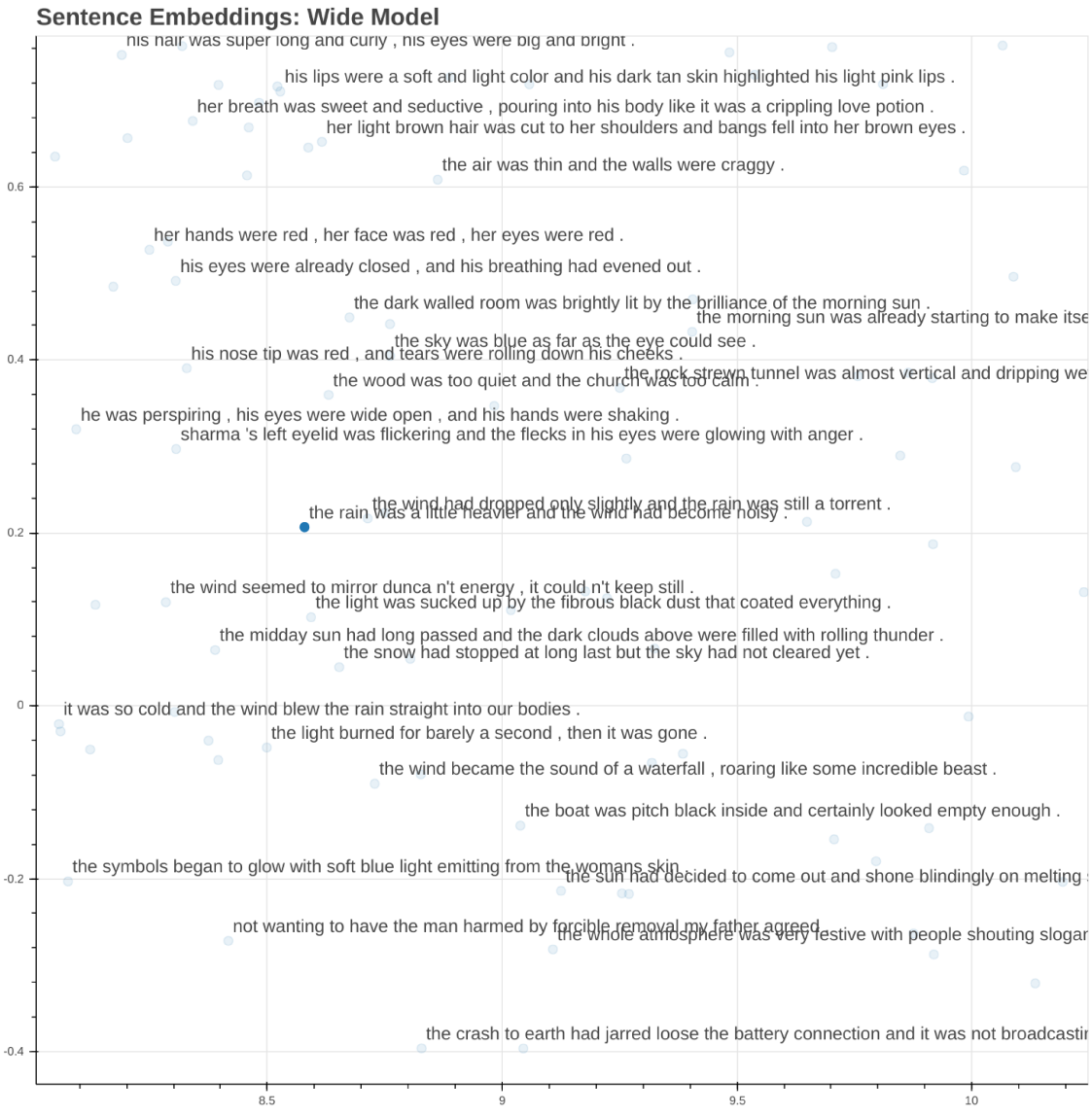

Figure 4.7: Showing the local embedding space near the sentence "the rain was a little heavier and the wind had become noisy." for the wide model.

semantically related sentences near each other. Figure 4.10 shows what sentences the WDN model groups near the sentence *"the rain was a little heavier and the wind had become noisy"*. Both the strictly deep model and the strictly wide model group this sentence with similar ones, but also intersperse many sentences that are not clearly related to descriptions of weather, such as descriptions of faces or of crying out in pain. The WDN model, on the other hand, has learned to to group exclusively descriptions of the sky and weather in this region.

Figure 4.8: Showing the local embedding space near the sentence "the rain was a little heavier and the wind had become noisy." for the deep model.

4.3 Autoencoding

Given that the model is able to learn semantic sentence representations, the rest of this chapter explores whether that representation can be used to generate paraphrases. In an unsupervised setting, where there are no explicit examples of target paraphrases, we need some way of training the model the produce a sentence with the same meaning, but with different words than those used in the input sentence. We start by simply fine tuning a model to autoencode the input sentence. That is, the

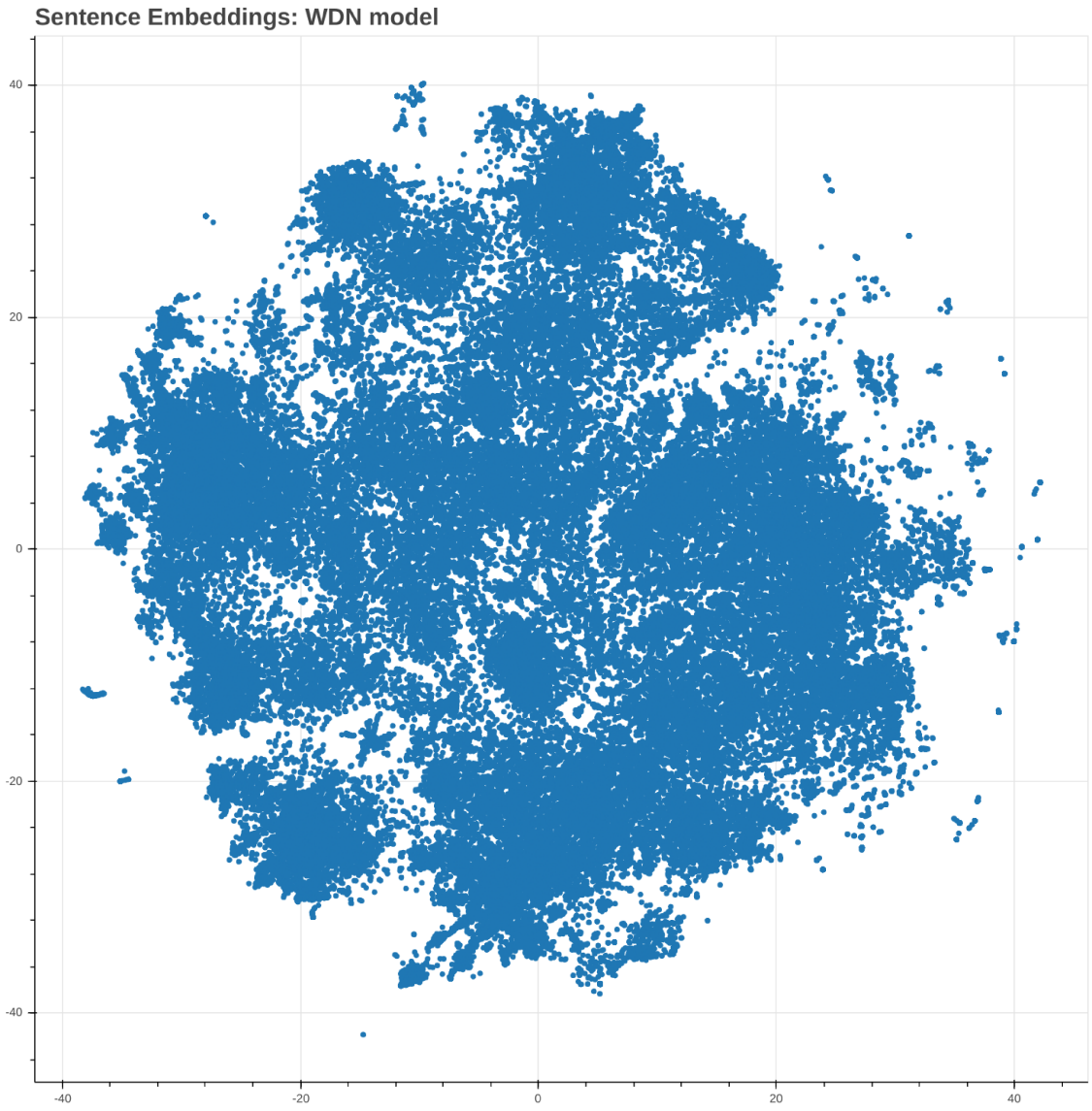

Figure 4.9: Showing the general structure of the sentence embeddings space for the WDN model.

language model trained in section 4.2 is fine tuned to reproduce the input sentence rather that attempt to predict the following sentence.

In order to prevent the model from simply copying the characters through and unlearning the semantic representations, all of the weights are frozen except those of the sentence decoder. The word encoder, sentence encoder, and word decoder weights are kept fixed. The motivation is to force the sentence decoder to learn a mapping from the semantic space of the sentence encoder's outputs to the semantic space of the

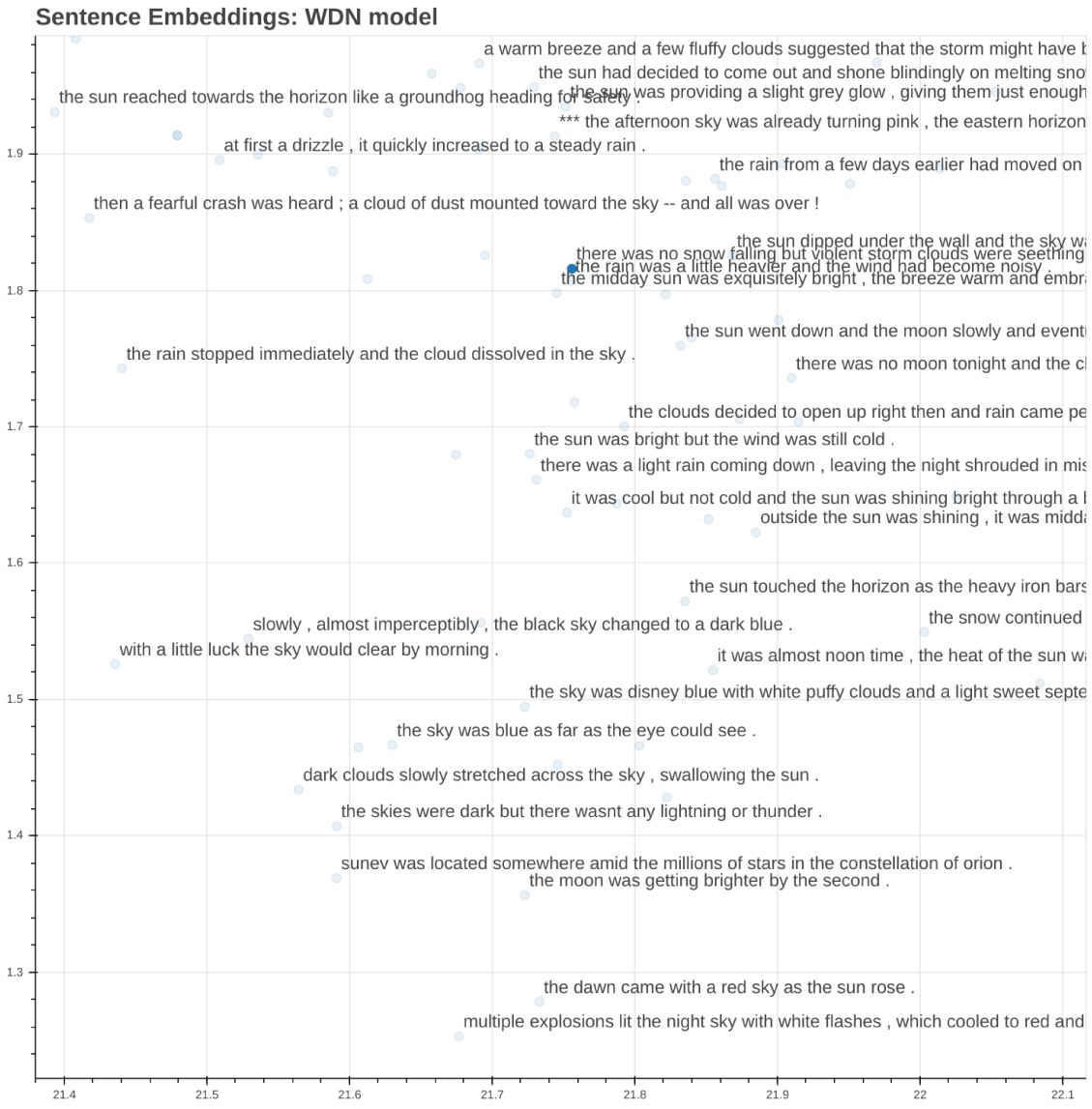

Figure 4.10: Showing the local embedding space near the sentence "the rain was a little heavier and the wind had become noisy." for the WDN model.

word decoder's inputs. For these experiments WDN from section 4.2 is used because it exhibited better semantic clustering.

During pretraining, the targets are composed of as many as the next 3 sentences, up to a maximum of 30 words. For autoencoding, however, the targets consist of only the input sentence. At the character level, the targets consists only of the characters on the current word, rather than the fixed length window of 20 characters described in section 4.1. The performance of the autoencoder under these restrictions is limited.

It is trained only for a short time, and relies on fixed weights learned during the pretraining phase. The reconstruction cost for a number of sentences is listed in table 4.2. The cost is the cross entropy as measured in nits per characters.

Table 4.2: Example sentences and the cost, in nits per character, of reconstructing them with the autoencoder.

Cost	Sentence
0.00	what do you mean ?
0.00	thank you.
0.00	what are you talking about ?
0.00	it could be anything .
0.00	whats wrong with him ?
0.00	i know where you come from .
0.00	is that what i think it is ?
0.00	she cleared her throat .
0.00	i need to get out of this room .
0.02	then the phone rang .
0.03	is that what you want me to believe ?
0.03	she looked me in the eye .
0.03	everything was going to happen fast .
0.04	i have to be dreaming .
0.05	this may take some time .
0.06	he did not know what was going to come next .
0.08	he pulled his phone out of his pocket and nodded his head
0.12	he asked with consternation in his voice .
0.14	i never doubt the man with the gun , the man replied .the police did not think anything at first .
0.19	i stared at the screen and it started to get blurry .
0.20	without another word , she walked out of the courtyard and through the front of the hotel .
0.24	the ringing of the phone brought him out of his thoughts and he got up and grabbed it off the wall .
0.47	metal clattered against stone as he crashed down to the gravel pathway .
0.63	simons dappled horse was gone , but to her surprise , she heard reynalds voice coming from the back of the wooden structure .
0.86	we sat outside , surrounded by lilacs , fending off the occasional mosquito , drinking that sweet drink the swedes call saftwhich ,
1.20	agnar battered a fast drum fill and then they were off ; a galloping double-bass drum beat supported scything guitar and bass ,

4.4 Generating Paraphrases

The purpose of training the autoencoder to reconstruct the input sentence is as a way achieving paraphrase generation in an entirely unsupervised framework. As shown in table 4.2, the reconstruction cost for short sentences with common words is often 0, which means that the input sentence is reproduced verbatim. For longer sentences, or sentences with less common words, the reconstruction cost is non-zero, meaning that the output sentence differs in some respect from the input. In some cases, this difference is the result of paraphrasing, where the decoder has altered the input sentence in a way that retains the meaning. In other cases, the difference between the input and output sentences reflects a change in meaning, sometimes even resulting in nonsense. The autoencoder therefore already generates paraphrases in some cases, and learns to do so entirely from reconstructing sentences embedded in a semantic space. On the other hand, the autoencoder frequently does not produce a paraphrase, and it is difficult to measure whether or not it has without manually inspecting it. Section 4.5 investigates some potential approaches to dealing with these challenges.

4.5 Non-deterministic paraphrase generation

In order to increase the chances of generating a valid paraphrase, two simple approaches are investigated. The first is simply to sample from the output layer of the last network rather than selecting the most likely character. The output layer of the decoder represents a probability distribution over all possible characters. This distribution can be sampled at each character, resulting in a non-deterministic output sequence, and doing so can sometimes produce a paraphrase where choosing the most likely output would not. The other approach is to perturb the encoded sentence representation by sampling from a random distribution and adding the result. For a given source sentence, the sentence encoder produces a vector representation ($\boldsymbol{s}_*^{(l)}$) at each layer, where $\boldsymbol{l}$ indexes the layer. This representation is then perturbed to produce $\boldsymbol{s}_*^{(noisy:l)}$ according to the following formula where f^σ computes the standard deviation over the values of the input vector, $\boldsymbol{w}$ is a scaling factor, and $\mathcal{N}$ represents the normal distribution which is sampled to produce $\boldsymbol{n}_*^{(l)}$:

$$\boldsymbol{v} = \boldsymbol{w} f^\sigma(\boldsymbol{s}_*^{(l)})$$

$$\boldsymbol{n}_{*}^{(l)} \sim \mathcal{N}(0, \boldsymbol{v}^2)$$

$$\boldsymbol{s}_{s*}^{(noisy:l)} = \boldsymbol{s}_{s*}^{(l)} + \boldsymbol{n}_{*}^{(l)}$$

The variance of the perturbation is proportional to the variance of the values in the sentence encoding vector. This is simply a heuristic for choosing an appropriate amount of noise for any given sentence's representation.

While sampling the output layer or perturbing the sentence encoding can sometimes produce a paraphrase where the model would otherwise not have, these techniques are non-deterministic and the results vary from one run to another. Performing multiple runs with a variety of perturbation levels or softmax temperatures can increase the chances of producing a good paraphrase, but the ability to assess the quality of the results is necessary to separate good results from bad ones. It is possible for a human to make such an assessment, but an automated approach would clearly be better. A good paraphrase should use different words than the source sentence, and potentially change the structure of the sentence as well. In this sense the output sentence should have a high reconstruction cost with respect to the input. A good paraphrase should also retain the meaning of the original sentence, however, which is more important and much more difficult to measure. As an approximation, we take advantage of the fact that the sentence encoder embeds sentences in a semantic space. By taking a generated candidate paraphrase and re-encoding, its proximity to the original sentence can be measured in the embedding space. This technique can be used to compare multiple candidates from successive, non-deterministic runs, and to choose the best result.

4.6 Discussion

Table 4.3 contains a number of examples of source sentences and the resulting candidate paraphrases ranked by their proximity to the source sentence's embedding. For each source sentence, 40 candidate paraphrases are generated, choosing a random value for the softmax temperature and the weight value $\boldsymbol{w}$ on the perturbation. Both values are sampled from a uniform distribution between 0 and 1. The table shows the four closest candidates and an additional four which are randomly chosen. Where there are fewer examples, it is because those are the only unique candidates produced.

The metric used is Euclidean distance, and although it provides a rough estimate of semantic similarity, the best paraphrase candidate is not always the closest.

The model struggles to accurately paraphrase longer sentences, but still produces candidates that seem to be at least related to the source sentence. For example, the sentence *"the oaks in front of them rustled their budding branches reaching up towards the endless sky"* is reproduced as *"the chirps of the trees moving up their steps pressed them back on to stars streaming deep below"*. While the exact meaning of the original sentence is lost, there are a number of qualities that have been preserved. For example both sentences refer to sounds produced by trees, and whereas the original describes branches reaching up to the sky, the reconstruction describes the trees pressing down onto the stars.

The model also frequently substitutes proper nouns, or mixes up body parts, numbers, colours, and times. For example the sentence *"devontay studied the ragged map in his hands"* is reproduced in one of the runs as *"torus studied the map in his lower lip"*. While this is clearly a failed paraphrase, it at least shows that the model is able to represent that the character has a name, and that the character has a map in one of their body parts. If the representation for 'hands' and 'lower lip' are near each other, which we would expect for a semantic encoding, it could be that this mix-up is a result of an unfortunately sampled perturbation.

There are also a number of instances where a good paraphrase is produced, but it is not ranked by the distance metric as the most similar to the original. For example *"she didnt care about having to know how things were"* more closely matches *"she didnt care about knowing how things really were"* than does the higher ranked ouput: *"she didnt care about things what were really knowing"*. Similarly, the closest ranked match to *"have you ever been in it ?"* is *"have you ever been some in it ? "* when the lower ranked *"have you ever been into it ?"* is clearly better.

Still, there a number of instances where the model works well, and is even able to produce multiple versions of sensible paraphrases for the same sentence. For example, the sentence *"a plan slowly began to take shape in his mind"* is reproduced as *"a sense of plan began to fill in his mind"*, but also as *"a plan began to flow in his mind"*, and the lower ranked attempt of *"the plan began to take a strong image out in his head"*. The model is able to creatively re-express the concept of something "taking shape

in the mind" as "filling the mind", "flowing in the mind" , and "an image in the head". This shows that language modeling does have the capacity to generate and decode semantic representations. Clearly one of the major limitations of this work is the lack of control over the generated sequence. By adding noise and sampling the output layer, a large amount of variability is introduced to the outputs which can hinder the ability to produce good paraphrases. Whereas this work constitutes a good proof of concept, further research is warranted to better quantify and improve the performance.

Table 4.3: Examples of generated candidate paraphrases. The original sentence is shaded grey, and its distance (in the encoding space) to each generated sentence is shown in the left column.

the line went silent .	
0.118	the line went quiet .
0.132	the low line went silent .
0.245	the list went silent .
" what are you going to say ?	
0.134	" what do you have to say ?
0.164	" what are you saying ?
" are you all right ? "	
0.126	" you all right ? "
0.140	" you are all right ? "
have you ever been in it ?	
0.079	have you ever been some in it ?
0.088	have you ever been into it ?
0.140	have you ever gotten in it ?
i must have done it wrong .	
0.108	i must have taken it wrong .
0.119	i must have gotten it wrong .
0.122	i must have given it wrong .
is he really that upset ?	
0.073	is he really upset that ?
0.197	is he really that angry ?
its fine , go and see him .	
0.076	its fine , leave and see him .
0.088	its good , go and see him .
0.126	its okay , go and see him .

0.088 its good , go and see him .

what the hell is your problem ?

0.044 what the hell are your problem ?

0.102 what the hell is your wrong ?

0.116 what the hell is your point ?

0.146 what the hell are you doing ?

0.044 what the hell are your problem ?

he sniggers .

0.137 he snorts .

0.141 he smirks .

0.157 he scoffs .

0.141 he smirks .

0.178 he laughs .

0.222 he snickers .

0.230 he sniffs .

i hear youre going home .

0.068 i hear youre getting home .

0.085 i hear youre coming home .

0.133 i heard youre going home .

0.276 i get you hearing home .

she knew she was in trouble tonight .

0.065 she knew she was in trouble today .

0.073 she knew she was in trouble tomorrow .

0.132 she knew she was in trouble yet .

0.198 she knew she was too in touch tonight .

0.215 she knew she was in work today .

he got out , and tried to go straight .

0.066 he got down , and tried to go out .

0.067 he got out , and tried to go .

0.072 he got up , and tried to go out .

0.072 he got up , and tried to go at once .

0.072 he got up , and tried to go out .

0.083 he got up , and tried to go forward .

0.083 he got out , and tried to go away .

0.136 he got up , and tried to go out slowly .

“ how long have you known ? ”

0.078 “ and how long have you known ? ”

0.083 “ how long have you known on your ? ”

0.127 “ how long you known ? ”

0.133 “ how long have you been ? ”
0.154 “ how long have i known you ? ”
0.168 “ how long have you seen ? ”
0.204 “ interesting how long have you been ? ”

his expression did not change , but something in his eyes did .
0.061 his expression did not change , but his eyes did it in .
0.064 his expression did not go , but something did in his eyes .
0.069 his expression did not change , but it did his eyes in .
0.070 his expression did not change , but it did in his eyes .
0.097 his expression did not heat , but something did on his eyes .
0.121 his expression did not change , but that did seem to change .
0.138 his expression did not help , but the eyes did it on his eyes .
0.213 his expression did not seem , because the light did up .

and it was now much more specific in what it was looking for .
0.084 and it was far more beneficial in what it was looking for now .
0.085 and it was more interesting in what it was looking for now .
0.086 and it was more certain what it was looking for in now .
0.087 and it was more plausible for what it was looking at now .
0.100 and it was more completely full in what it was looking for .
0.123 and it was more than what it was looking for in more surprise .
0.179 and it was more pitiful for what it was looking in now .
0.183 and now it was full in what approaching look for most .

was lying on the ground .
0.150 was laying on the ground .
0.171 was leaning on the ground .
0.191 was lies on the ground .

she didnt care about knowing how things really were .
0.056 she didnt care about things what were really knowing .
0.074 she didnt care about things that were really knowing .
0.075 she didnt care about how things were really .
0.083 she didnt care about having to know how things were .
0.075 she didnt care about how things were really .
0.103 she didnt care about really things what were .
0.137 she didnt know how to change things just yet .
0.193 she wasnt happy about how things were .

she snorted in reply , and i laughed .
0.095 she laughed in annoyance , and i laughed .
0.096 she laughed in confusion , and i laughed .
0.100 she laughed in astonishment , and i laughed .

0.102 she laughed in reply , and i laughed .
0.121 she laughed in unison , and i laughed .
0.124 she laughed , and i laughed in response .
0.135 she laughed in turn , and i laughed .
0.269 i laughed in the soft , and she chuckled .

he just wanted to survive high school and get on with his life .
0.088 he just wanted to come on high school and save his life .
0.090 he just wanted to get out of school and stay alive with his life .
0.094 he just wanted to get out of his life and school and start with life .
0.103 he just wanted to get out of his life and school with life .
0.106 he just wanted to get on his life and school with life .
0.138 he just wanted to get out of his life and stay in college .
0.331 he just wanted to lose his class as and get married on teeth .
0.355 he just wanted to get together , go into his whole life and wait on his war .

stopping her in the water until he could move up beside her .
0.067 stopping her in the water until he could stand up beside her .
0.090 holding her up in the water until he could move her back in .
0.097 holding her up in the water until he could stop her .
0.101 catching her in the water until he could move up beside her .
0.154 walking until he could stop her in the water over her .
0.165 rubbing her arms until he could stop her in the water .
0.196 just behind him until he could stop her in the water .
0.233 reaching up behind him he could get her under water at the moment .

a plan slowly began to take shape in his mind .
0.127 a sense of plan began to fill in his mind .
0.137 a plan began to flow in his mind .
0.139 a long plan began to rise in his mind .
0.141 a plan began to rise in his mind .
0.183 the plan began to take a strong image out in his head .
0.195 a mind began to take off in his mind .
0.206 a new journey began to put in his mind .
0.213 a thought began to part moving in his mind .

two dead bloody bodies lying on the beach .
0.212 two dead bodies lying on the sand .
0.219 two dead bodies lying on the grass .
0.219 two bad bodies laying bleeding on the grass .
0.230 three bloody bodies lying on the ground .
0.259 two bodies lying on the ground dead .
0.304 two dead bodies lying on the tree .

0.422 two guys looked dead on the back of a drift .
0.468 four gunpowder dead looking on the ground .
red exclaimed .
0.151 red replied .
0.151 red announced .
0.193 red interrupted .
0.193 red interrupted .
0.199 yellow exclaimed .
0.524 blue exclaimed .
i parked my car across the street from her house .
0.031 i parked my car down the street from her house .
0.065 i parked my car in front of her house from the street .
0.077 i parked my car in the front of her house .
0.090 i parked my car within the alley from her house .
0.118 i drove my car from the street behind her house .
0.134 i drove my car across the hill from her house .
0.158 i drove my car by her across the street .
0.209 i drove my car from her house behind the car .
i 've just got a hunch that something is off .
0.098 i 've still got a hunch that something is out .
0.145 i 've got something that is a bit over .
0.145 i 've got a bad idea that is off .
0.149 i 've got a bad idea that is out .
0.149 i 've got a bad idea that is out .
0.193 i 've got that a bad idea out .
0.217 i 've got a bad thought where that 's going .
0.560 i 've got a book just finding that .
“ well , do n't let me slow you down , ” he said .
0.053 “ well , do n't let me down you slow , ” he said .
0.058 “ well , do n't let me down you close , ” he said .
0.066 “ well , do n't let me stop you down close , ” he said .
0.074 “ well , do n't let me get you close , ” he said .
0.058 “ well , do n't let me down you close , ” he said .
0.066 “ well , do n't let me stop you down close , ” he said .
0.321 “ do n't let me climb over , you said far , ” he said .
0.429 “ do n't let me down , you whispered quickly . ”
the oaks in front of them rustled their budding branches reaching up towards the endless sky .
0.199 the chirps of the trees moving up their steps pressed them back on to stars streaming deep below .

0.200 the trees pointing up at them in the cold breeze held their paces above deep red .

0.203 the trees standing up in front of them stretched their feet in the dark sky above their streams .

0.210 the trees stretching up in front of them the sun stretched toward their sharp surfaces .

0.247 the streams of the sun passed up them lining their feet toward the streaming sky in strange shadows .

0.259 the close round of the snow turned up to hurl their clouds at their supplies having lived here in rich arches .

0.279 the stones in the clouds passed up their tails staring up at them from below the high shades of sun .

0.284 the stones still up in their palms moving toward the north horizon covered them in green glow of air .

a young boy with freckles all over his face and short red hair answered , i can do a little , but i keep falling down .

0.168 a young boy with a black hair and a black beard and a black eyes continued , i can do all i can , but i get up straight .

0.171 the older man with a black hair and a black haired boy with a blue eye said , i can do all i can , but i start falling down .

0.180 a young boy with a cold blue hair and a blue eyes said , i can do all i can , but i keep going up slowly .

0.190 a little older boy in a red hair and a white hair seemed to grow completely pure and he answered , i can do , i just rush back .

0.180 a young boy with a cold blue hair and a blue eyes said , i can do all i can , but i keep going up slowly .

0.253 the old man with a brown hair and a brown beard and a feathered face , i believe i can go down , say i all but is running on head .

0.264 my brother a long , short brown haired boy with a wide blue eyes and a smile was buried on his face , i can go and i do .

0.408 his old man had a beard and a tongue behind the girl who continued to jab his eyes and told me , i can just run up , but i stay too slight .

at last , an hour after they started , the entire heart had taken form , though the muscles were still much less than full strength .

0.216 at least , the pain had passed , after they were finished , the feelings were long enough , even with a few hundred years of relaxation .

0.219 at last , the grip on their bodies had taken , while the confidence lasted a few hours , even though some life was now uncomfortable .

0.225 at last , the hard rest had set , the pain from their faces were now drained , even though they were also a long heart attack .

0.230 after a few minutes , the pain had been destroyed , even though , their heart was almost a hundred feet from the long , more sense to be taken .

0.219 at last , the grip on their bodies had taken , while the confidence lasted a few hours , even though some life was now uncomfortable .

0.266 when they reached , the heat had started , deep in not enough , the most perhaps two years were for a long moment just back from awkward fear .

0.283 the thought rose , after a few more minutes , their bodies were forgotten , even when the pleasure had been so long poisoned at last .

0.351 after they had fallen , the sensation was long , long , but the life went on , even the deepest tears were coming more than one they had far more access .

they are hiding but our look out on the roof can see them .

0.136 they are looking out at the roof but we can see their way .

0.169 they are looking out our windows but can see them on the ground .

0.170 they are looking out the window but can see them in our path .

0.193 they are looking out the window but can we see on our feet .

0.235 we are looking out the window but can see their back .

0.273 we are out in a trap but can see their look their door .

0.337 we are out on their street but can see the glass they have on .

0.345 they are only looking out onto the trees but we can see our place .

it was mostly dark with only indoor lantern lights .

0.170 it was very open dark almost with thick lights .

0.177 it was usually hot just in dark shutters with huge light .

0.184 it was also pretty dark with dark lights .

0.188 it was already dark with plenty of dark lights .

0.207 it was only dark with perfectly dark lights .

0.282 it was always dark in thick windows with still shadows .

0.286 it was always perfect dark with tall concrete through cold windows .

0.354 it was almost dark and even with shades colored cameras .

he had only drifted , feeling untethered from his body , a balloon of pure thought rising away from its string .

0.205 he had felt , exactly dreaded , a skin flying in from a fire dispersing at his brain without its turn of its appearance .

0.213 he had thought , a chaotic emotion mixed outwardly , forming away from his body , it splashed like a deer at forming held onto leaving .

0.215 he had not missed , a hand shifted from his body , an odd new train filling its thoughts at flying .

0.224 he had felt , dreaming of a sharp , strong blow had stopped from his body by a stream at its end .

0.228 he had felt it , his body , a hot pulse of heat that had never been pushed from a creature from distracting him from awe .

0.247 he had felt an anger , a wave of soft , dream had been brought from its right to fall from his arm .

0.257 he had felt his body continued , feeling a pain , the understatement was flowing straight from flame to from its destruction .

0.262 he had felt a thunder , only its source , with fear of realizing he was still swimming from being pulled at below .

phil hammer raised his mug of frothy beer .

0.215 phil straightened his cup of coffee beer .

0.220 phil started his cup of coffee sticking in steel .

0.231 phil stuck his cup of cup cold stale .

0.236 phil brought his cup of glass wine sharply .

0.237 phil pulled his fork of cup ale thumbed .

0.331 paul took his cup of coffee bar short black spit .

0.334 joe thud sank his black bottle of tea .

0.709 jo rose cupping his arm of beetle cup .

goosebumps covered his body , which shivered violently as it tried to warm itself .

0.186 parts smelled softly , which it took to breathe which savoured his gut .

0.191 tender murmurs ran to his body , as it anticipated that it only smelled chilly .

0.204 weak , his right aroma hung as it touched his limbs , which was a full pain teasingly .

0.214 sweat spilled over his body , which it sounded strong enough to shiver its stinging sensations .

0.232 laughter filled his veins , which it took softly to stay wet as it unfamiliar was weak .

0.260 gritted teeth , his arms were so cold that it pressed to feel itself .

0.265 tears grew painful , as it floated straight to his face which felt oddly wet .

0.305 shaking his hot flesh inside it , which took almost ease to shake the sandal felt weaker .

at half a mile wide , it resembled a small , opal-colored moon hovering just above the grass .

0.196 at a back , it tapered up with a small white horizon , the bright trees still contrasted completely shaped .

0.220 at a distance , it appeared a curved white dust hill , settling just over the third shadow .

0.234 at the edge , it appeared a small , cast black land at a stone thirty-five miles ahead .

0.235 in the distance , it appeared a fairly black , blue cloud , a tree surrounded by thick brown sky .

0.241 at the base , it stood at a small gray brown stream , a small darkness expecting every view .

0.257 at a moment , it an opening large white stream hung at the centre , leading just a dim white .

0.281 at a slow , it was a small , streaming street that showed a small blue sky stretching at the edge .

0.288 at one end , the cluster gazed at a red , other-side dirt now tracking a settled moonlight .

im ordering the tankers to swing around to the far side of the gg ,

0.323 im holding the bay of the command to the base for batteries to fly along the south ,

0.324 im trying to stride the ship to the top of the companies from the 100 meters ,

0.343 im tending to the ships to slide down the top of the tracks to c.

0.348 im trying to clamber the american to the next short of the horizon in the production ships ,

0.323 im holding the bay of the command to the base for batteries to fly along the south ,

0.382 im on the side of the ships to get to the top lines along the area ,

0.386 with my owner im positioned to the nest of the navy to throw down the top ,

0.391 im looking forward to the units of the base to roll across the front ,

devontay studied the ragged map in his hands .

0.386 torus studied the map , his fingers cradling in his knee .

0.398 torus studied the map in his lower lip .

0.416 torus studied the map of the hills hand in his head .

0.418 torus studied the map in his fingers .

0.416 torus studied the map of the hills hand in his head .

0.652 official river stared at the long metal videos .

0.684 one ring looked glanced at his coming trains in the shaft .

0.689 delaney studied the map in his thin fingers .

star dawning is an inspiring story of individual redemption and the restorative power of family .

0.260 star is a secret of the soul of ancient and self-conscious students desiring death .

0.277 star is a story of the service of soul and desire at souls age .

0.286 star is a secret of a self-destructive story of love and consciousness considering their souls .

0.296 internet is a perfect conversation of fear of soul and the constant war of children .

0.348 star is a secret death of the powerful and self-suffering story of normal people .

0.408 citizen (attempt was a modesty of a life is the childhood stories of body character and artthe losing class .

0.546 stout is a topic of stories of wisdom and love convertible the entirely depraved death .

0.583 douglas is a dream of the infested being of many quick treatment and self-imposed sense of human secret .

not even louise could have spotted me .

0.338 not even lucy could have saw me .

0.347 not even hernie could have seen me .

0.370 not even helen could have spotted me .

0.387 not even the lady could have saw me .

0.347 not even hernie could have seen me .

0.496 not evangelically ella could have seen me .

0.564 not even sarah could have saw me .

0.596 not daniel could have scoured me .

he easily memorized the look of the kings , queens , jacks , and aces .

0.367 he always recognized the look of the kings , heinekens , kriegmans , and concubines .

0.385 he also looked the way of the kings , strangers , blue , and silver .

0.391 he expected the looks of the kings , salad , and fathers , thomas .

0.399 he also gave the look of the grands , servants , sons , and mates .

0.385 he also looked the way of the kings , strangers , blue , and silver .

0.554 he expected the tears and shoulder of the guard , ezekiels , dragons , markets .

0.629 he cared enough of the royal lines , dardan , the guys , and gowns .

0.765 he remembered the looks of the kings , marmel , and boys , beings .

Chapter 5

Discussion

5.1 Overview

This work investigates language modeling, its effectiveness at producing semantic representations at multiple levels of hierarchy, and the ability to use these representations to generate paraphrases in a completely unsupervised setting with no examples of paraphrases. The architectures investigated include the SARAh, a novel architecture that uses an internal attention mechanism similar to that of Transformers, but which implicitly and recurrently encodes sequence positions. It is shown that language models composed of a variety of components are capable of building representations of words from characters, sentences from those word representations, and then back down through words to character outputs. The ability to capture the stylistic attributes of a given author is explored, and we find that it is possible to fine tune a language model to generate text that more closely matches the style of a given author, while taking advantage of having learned on a larger dataset.

The ability to produce semantic representations at the sentence level is explored briefly by looking at part of the sentence embedding space and seeing what sentences are encoded near each other. This shows that the encoder portion of the model is capable of mapping from characters to a semantic feature space. In order to assess how well the decoder is able to make use of this semantic encoding, this work investigates the ability to produce paraphrases from the encoder's representation. By freezing many of the encoder weights trained during language modeling (as well as those of the word decoder) and fine tuning the sentence decoder's weights to autoencode the input, it's possible to generate sentences that are semantically related to the input sentence. Multiple attempts can be made at generating the output sentence by adding varying amounts of noise to the sentence representation, or by sampling the outputs. In some cases, the result is a paraphrase, in other cases it is only loosely related, and other times it is not at all related. The similarity of the generated sentence to the

original can be approximated by re-encoding the outputs and comparing its proximity to the original in the semantic embeddings space.

The ability to generate paraphrases with no training examples shows that language modeling allows deep neural networks to map text to a feature space that captures semantic information about that text. However, generating paraphrases can also be useful. For example, a paraphrase could serve as a form of inspiration for authors, and could help to alleviate writer's block. By rewording a sentence, a paraphrase can also help to assess the clarity of that sentence: if a competent paraphrase generator does not accurately capture the intended meaning of the sentence, that meaning may not be clear in the original sentence. Another related application for paraphrases is text simplification, in which text is reworded in order to improve quality and make it more concise. In order to do this, the paraphrase generator would need be trained to produce sentences that are shorter and simpler than its inputs. One possible avenue for this is simply to fine tune the decoder with supervised examples. Another option could be to use stylistic fine tuning, similar to what was described in section 3.3.

Style transfer has been successfully applied in image processing, where the stylistic attributes of a particular painting can be separated from the content and applied to a new image.[13] Only recently has progress been made at developing various methods for achieving the related goal of linguistic style transfer. [26] [51] [38] [12] However, these works mostly focus on the sentiment of the generated text, converting a negative statement to a positive one or vice-versa. This does not constitute a paraphrase because the meaning of the sentence has changed, and is distinct from capturing the less easily defined concept of author style. The work presented here provides a road-map for transferring author style by combining the ability to generate paraphrases with the concept of stylistic fine tuning.

Another avenue for future investigation regarding linguistic style is to explicitly condition the model to an author or other domain by providing a signal that identifies that domain during training. A simple implementation of this signal would be to attach embeddings or one-hot encodings of the given author to the inputs during training. This is similar to follow up work done on music generation[39] in which a model is conditioned on the year of a composer's birth during training, thereby allowing the style of the generated music to be modulated to resemble the style of a

given period.

The quality of the paraphrase generator also needs to be improved to allow paraphrases of longer and more complex sentences. The main challenges for this are that the sentence representation must be packed into a fixed length representation, and that many of the model's weights need to be frozen during autoencoding to force it to use the learned semantic representations. The first problem can be mitigated by using larger state sizes in the network, but this comes at the cost of more parameters and longer training time. Both challenges, however, could be overcome by using a small set of supervised training examples. Once the model is pretrained using a very large unsupervised text corpus, it may be possible to use a relatively small, high quality training set to fine tune it. This would allow the possibility of providing the decoder with the variable length, per-word outputs of the encoder. This would not encourage the decoder to simply copy the input sentence verbatim since the targets in this scenario would differ from the inputs. For the same reason, such a training set has the potential of allowing for all of the model's weights to be fine tuned, not just those of the sentence decoder. An ideal training set would contain paraphrases that differ significantly from the original sentence in terms of wording and structure.

Despite these challenges, generating paraphrases, even simple ones, with no supervised training examples showcases the ability of DNNs to discover meaningful relationships in data. The ability to map text to a semantic space, and to decode from that space back to text is crucial for producing any system that deals intelligently with language. Unsupervised paraphrasing demonstrates that DNNs can learn to do this implicitly, and need only tap into the easily available, nearly limitless sources of raw text to do it.

Bibliography

[1] Basant Agarwal, Heri Ramampiaro, Helge Langseth, and Massimiliano Ruocco. A deep network model for paraphrase detection in short text messages. *CoRR*, abs/1712.02820, 2017.

[2] Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. *CoRR*, abs/1710.11041, 2017.

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[4] Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python*. OReilly Media Inc., 2009.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[6] Ignacio Cases, Minh-Thang Luong, and Christopher Potts. On the effective use of pretraining for natural language inference. *CoRR*, abs/1710.02076, 2017.

[7] Jianpeng Cheng and Dimitri Kartsaklis. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. *CoRR*, abs/1508.02354, 2015.

[8] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv e-prints*, October 2018.

[10] William Dolan, Chris Quirk, Chris Brockett, and Bill Dolan. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. International Conference on Computational Linguistics, August 2004.

[11] Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. Structural representations for learning relations between pairs of texts. In *ACL (1)*, pages 1003–1013. The Association for Computer Linguistics, 2015.

[12] Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. Style transfer in text: Exploration and evaluation. *CoRR*, abs/1711.06861, 2017.

[13] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[14] Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. A deep generative framework for paraphrase generation. *CoRR*, abs/1709.05074, 2017.

[15] Hua He, Kevin Gimpel, and Jimmy J. Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In Llus Mrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP*, pages 1576–1586. The Association for Computational Linguistics, 2015.

[16] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.

[17] Jeremy Howard and Sebastian Ruder. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146, 2018.

[18] Yangfeng Ji and Jacob Eisenstein. Discriminative improvements to distributional sentence similarity. In *EMNLP*, pages 891–896. ACL, 2013.

[19] Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *CoRR*, abs/1602.02410, 2016.

[20] Kazuya Kawakami, Chris Dyer, and Phil Blunsom. Learning to create and reuse words in open-vocabulary neural language modeling. *CoRR*, abs/1704.06986, 2017.

[21] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. *CoRR*, abs/1508.06615, 2015.

[22] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *CoRR*, abs/1506.06726, 2015.

[23] Shibamouli Lahiri. Complexity of Word Collocation Networks: A Preliminary Structural Analysis. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[24] Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Phrase-based & neural unsupervised machine translation. *CoRR*, abs/1804.07755, 2018.

[25] Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. A continuously growing dataset of sentential paraphrases. *CoRR*, abs/1708.00391, 2017.

[26] Juncen Li, Robin Jia, He He, and Percy Liang. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *CoRR*, abs/1804.06437, 2018.

[27] Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. Paraphrase generation with deep reinforcement learning. *CoRR*, abs/1711.00279, 2017.

[28] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004.

[29] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[30] Minh-Thang Luong and Christopher D. Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. *CoRR*, abs/1604.00788, 2016.

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[33] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[34] Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek V. Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. Neural paraphrase generation with stacked residual LSTM networks. *CoRR*, abs/1610.03098, 2016.

[35] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[36] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[37] Vasile Rus and Mihai Lintean. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162. Association for Computational Linguistics, 2012.

[38] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. Style transfer from non-parallel text by cross-alignment. *CoRR*, abs/1705.09655, 2017.

[39] Ian Simon and Sageev Oore. Performance rnn: Generating music with expressive timing and dynamics. https://magenta.tensorflow.org/performance-rnn, 2017.

[40] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[41] Michael Traynor and Thomas Trappenberg. Words are not temporal sequences of characters. In *2018 International Joint Conference on Neural Networks (IJCNN)*, July 2018.

[42] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. volume 9, pages 2579–2605, 2008.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[45] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. Sentence similarity learning by lexical decomposition and composition. *CoRR*, abs/1602.07019, 2016.

[46] Ludwig Wittgenstein. *Philosophical Investigations*. Basil Blackwell, Oxford, 1953.

[47] Ludwig Wittgenstein. *The Blue and Brown Books*. Harper & Row, 1958.

[48] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[49] Wei Xu, Alan Ritter, and Ralph Grishman. Gathering and generating paraphrases from twitter with application to normalization. In *BUCC@ACL*, 2013.

[50] Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. Unsupervised neural machine translation with weight sharing. *CoRR*, abs/1804.09057, 2018.

[51] Zichao Yang, Zhiting Hu, Chris Dyer, Eric P. Xing, and Taylor Berg-Kirkpatrick. Unsupervised text style transfer using language models as discriminators. *CoRR*, abs/1805.11749, 2018.

[52] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *arXiv preprint arXiv:1506.06724*, 2015.