Article Title Generation Using Recurrent Neural Networks With Attention

Dominik Matić[1]*
C. L. F. Nimbus[2]
[1]Faculty of Electrical Engineering and Computing, Zagreb, Croatia
[2]Šibenik, Croatia

*Corresponding author: Dominik Matić; dominik.matic@outlook.com

Abstract

Recurrent neural networks, usually supported by attention mechanisms, can reliably solve many different problems such as text summarization and language translation. This paper experiments with limits of such networks by introducing a new type of problem: generating a title for an article given the abstract. This problem was chosen because it is similar enough to an already well-studied problem of text summarization, and also distinctly different from it in the fact that article titles don't necessarily have to be descriptive of the article in an obvious way. The scope of this paper includes an overview of related models and techniques used on similar problems, descriptions of how data was acquired and prepared, how the model was implemented and trained as well as results analysis and discussion.

Keywords: recurrent neural network; attention; machine learning; natural language processing

1 Introduction

The idea behind recurrent neural networks is to be able to process sequential data of various and unspecified lengths to perform specific tasks. This is done by keeping a hidden state and updating it with each consecutive data input hoping it will remember the necessary information about the sequence to be able to perform the task at hand. There are a couple of ways an RNN can map an input to an output, all of which are shown below.

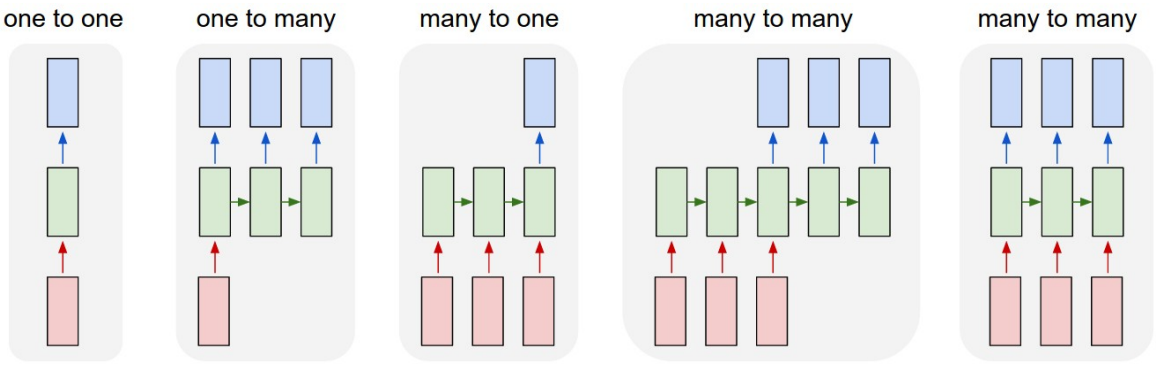

Figure 1: Types of RNN input to output mappings (Karpathy, 2015)

In each of these cases one column represents one time step of an RNN where the red, green and blue rectangles represent the input, the hidden state, and the output, respectively. The arrows indicate the flow of information within the model from which is clear that the hidden state evolves over time and propagates information about the sequence to the outputs.

The data being processed could be anything that contains a 'time' dimension such as text, audio, DNA sequences, or financial market data. Consequently, RNNs are a very popular choice when dealing with NLP[1]

[1]Natural Language Processing

problems most notable of which are language translation, text summarization, and part-of-speech tagging. The basline RNN model architecture in those cases is an attentional encoder-decoder architecture proposed by Bahdanau, Cho, and Bengio (2014) which added an attention mechanism to an already existing encoder-decoder architecture (Cho, van Merriënboer, et al., 2014).

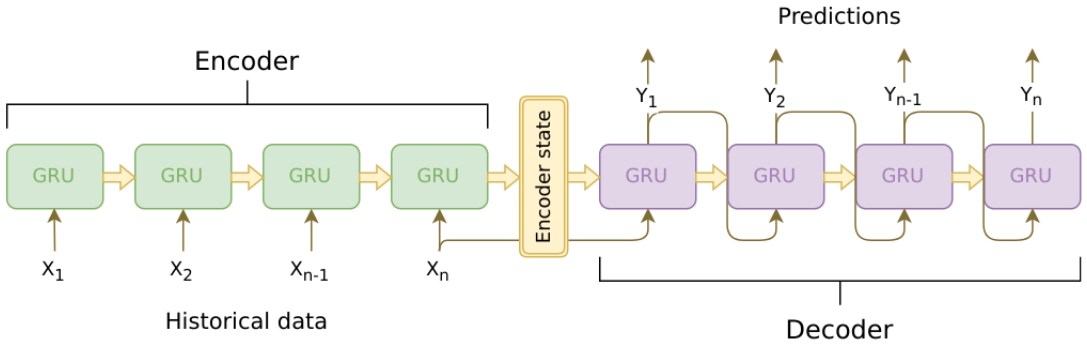

Figure 2: RNN encoder-decoder architecture, source: Joseph Eddy

With everything said in mind, an RNN seems like an interesting candidate for experimenting with different tasks and measuring performance. In this paper, an attentional RNN encoder-decoder model is implemented and tasked with generating a title for an article given its abstract. This was chosen specifically because the task is comparable to an already well-studied problem of text summarization and because it is different in a way that articles don't necessarily have to summarize the article itself in an obvious way.

2 Related work

In recent years, when it comes to NLP related tasks, RNN models have been outperformed by transformer models (Vaswani et al., 2017). Most notable of which are Google's BERT (Devlin, Chang, Lee, & Toutanova, 2018) which works best on discriminative tasks, OpenAI's GPT-3 (Brown et al., 2020) designed for generative tasks, and Facebook's BART (Lewis et al., 2019) which combines the concepts from the previous two. While they're very robust and seemingly able to perform any language specific task, such as the one this paper is trying to achieve, they are very large, can take relatively long time to fine-tune, and simply aren't always necessary for every task. That being said, RNNs are still being used either in conjunction with transformers or in standalone models where transformers aren't necessary or don't perform as well. In the case of title generation, while certainly able to perform the task, they might not be necessary.

Title generation is a task very similar to abstractive[2] text summarization already focused on by Nallapati, Zhou, santos, Gulcehre, and Xiang (2016) where they use a model similar to the one used in this paper to scan an entire document with the goal of generating an abstractive summarization in the form of a few sentences. Their model is based on the aforementioned attentional RNN encoder-decoder model proposed by Bahdanau et al. (2014) with additional upgrades. They implement a feature rich encoder which takes into account linguistic features of the word alongside the word itself, a switching generator-pointer mechanism which helps to deal with words not present in the dictionary, as well as a hierarchical attention mechanism which modifies the attention to work on both word and sentence level.

Additionally, there have been many proposed architectures and alternative approaches which improved upon the results achieved by Nallapati et al. One such proposal came from Li, Lam, Bing, and Wang (2017) whose model borrowed concepts from variational auto-encoders introducing their generative latent structure to its architecture which improved results. Other approaches include those using convolutional neural networks (Zhang, Li, Wang, Fang, & Xiao, 2019) and generative adversarial networks (Yang et al., 2021). An overview of these papers, as well as some others, can be read in a study by Yadav, Desai, and Yadav (2022).

Generally, most of these improvements exist because the task of generating an abstractive text summarization has to deal with a very large input size, whereas our model only has to deal with abstracts, which are much shorter. Because of this, our model resembles the one proposed by Bahdanau (Bahdanau et al., 2014).

[2] As opposed to extractive, which extracts key sentences from the document instead of generating new ones.

3 Methodology

3.1 Dataset Description

The entire dataset was acquired from arXiv.org using Python's arXiv API querying the following keywords:

science, technology, biology, computer science, medicine, machine learning, equation, deep learning, distribution, learning, generation, model, image, graph, estimation, tree, molecule, quantum, genetic, computing, dna, engineering, data, study, advanced, astronomy, astrophysics, atom, beaker, biochemistry, chemistry, botany, cell, chemical, climate, control, electricity, element, energy, evolution, fossil, geology, geophysics, gravity, hypothesis, immunology, magnet, matter, meteorology, motion, organism, observe, phase, particle, physics, quantum mechanics, research, radiology, temperature, theory, tissue, variable, variational, volume, weather, zoology

Bias could exist in the keyword selection since they were chosen mostly arbitrarily and weren't given much thought. Nevertheless, 187538 unique title-abstract pairs were acquired in this way. Some of the entries from the dataset are shown below, abstracts were shortened for brevity.

Title	Abstract
The Science of al-Biruni	Al-Biruni (973-1048) was one of the greatest sc...
Embracing Data Science	Statistics is running the risk of appearing irr...
A New Task for the Philosophy of Science	This paper argues that philosophers of science ...
Science and Philosophy: A Love-Hate Relationship	In this paper I review the problematic relation...
Quantum Computer Systems for Scientific Discovery	The great promise of quantum computers comes wi...
A note on two notions of compliance	We establish a relation between two models of c...

Figure 3: Example entries in the dataset

Statistical analysis of the dataset concluded with the following results:

Word length

	Titles	Abstracts
Min	1	1
Max	47	619
Median	10	145
Mean	10.74488370356941	150.26933208203138
Standard deviation	4.100086118821517	64.23186043180522

Table 1: Dataset statistics

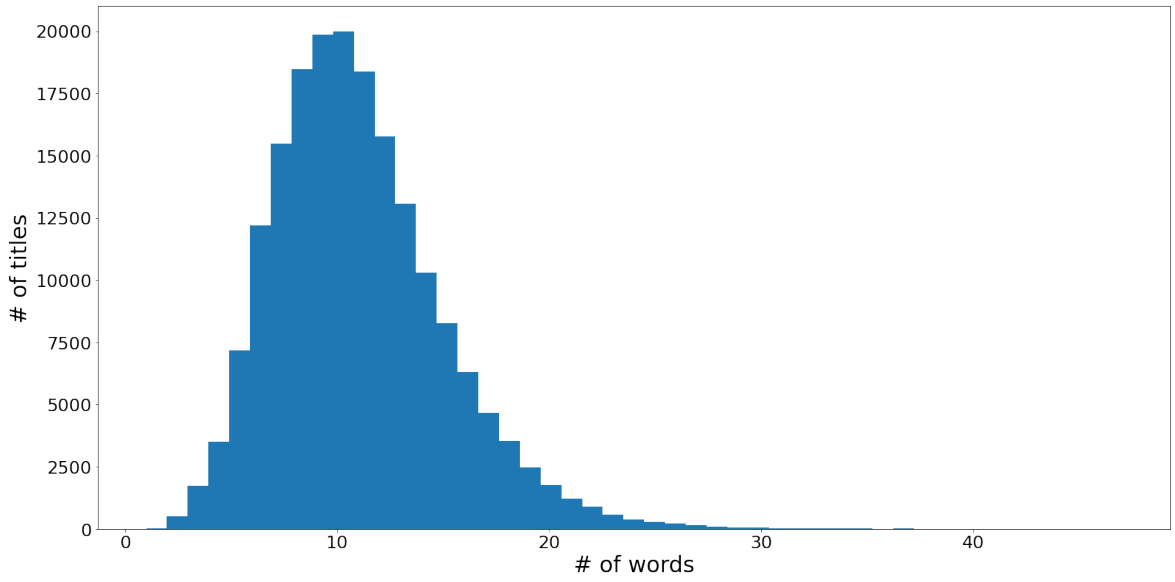

Figure 4: Histogram showing the title word length distribution in the dataset

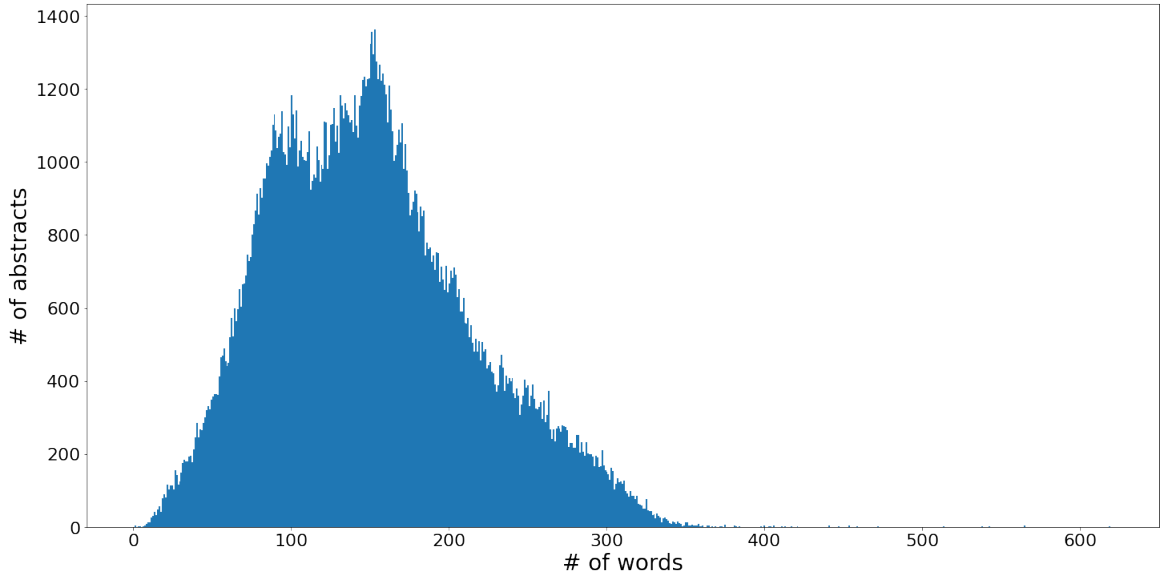

Figure 5: Histogram showing the abstract word length distribution in the dataset

3.2 Dataset Preprocessing

All of the words in the dataset were lowercased and tokenized. In tasks such as sentiment analysis, removing stop words generally improves performance, but in tasks such as this one context is essential and stop words greatly contribute to it, therefore **no** stop words were removed. Additionally, dataset entries whose titles exceeded more than 24 words were also removed prior to training because such entries would often cause out of memory issues. Alternatively, these issues could have been resolved by reducing the training batch size, but given that there are only 1036 entries that have titles whose word count exceeds 24, removing such entries seemed preferable to extending the time it takes to train the model. Dataset was then split into the traditional training, validation and test subsets in a ratio of $140000 : 23251 : 23251$.

The memory bottleneck of every sequence to sequence model is the size of its output dimension, in other words, its vocabulary. Because of this and because the model was trained on a GPU with only 4 GiB of memory, the vocabulary was constructed with words from the training subset only whose frequency is 10 or higher. During training, if the word was not in the vocabulary it was replaced with the unknown token.

Lastly, a word embedding matrix was partially initialized using GoogleNews-vectors-negative300 dataset which maps words to 300 dimensional vectors. Every word that wasn't in the dataset was initialized randomly following the normal distribution $\mathcal{N}(0, 1)$. All of the values in the embedding matrix were set as trainable.

3.3 Model Description

The model follows the attentional encoder-decoder architecture proposed by Bahdanau et al. (2014), the only difference being the use of Luong's attention (Luong, Pham, & Manning, 2015). The main idea is to use the encoder to process an entire input sequence into a hidden state which the decoder then uses to generate a new sequence. The issue with this approach is that it's generally very difficult to encode all of the necessary information within the hidden state. This is why we need attention.

Besides updating the hidden state after each step, an RNN cell also generates an output. The idea behind attention is to somehow utilize the encoder cell outputs in the generation process. This is done for each time step in the decoder by calculating a context tensor based on the current hidden state, current decoder input and all of the encoder outputs. Attention is then defined as a normalized vector which denotes how important an encoder output is for the current time step, which when multiplied with the encoder outputs results in the context tensor.

$$\mathbf{c}_i = \mathbf{a}_i \cdot \text{enc\_out} \tag{1}$$

This context tensor, along with the hidden state and an embedded input word, is then used as an input into the decoder's RNN cell to generate a new word. Luong expands on this and generalizes attention by adding an arbitrary scoring layer and using it on the context and the hidden state before using them as inputs to the cell, adding more flexibility. In this particular case, an arbitrary scoring layer was implemented by introducing additional parameters within the calculation and adding a *tanh* activation function.

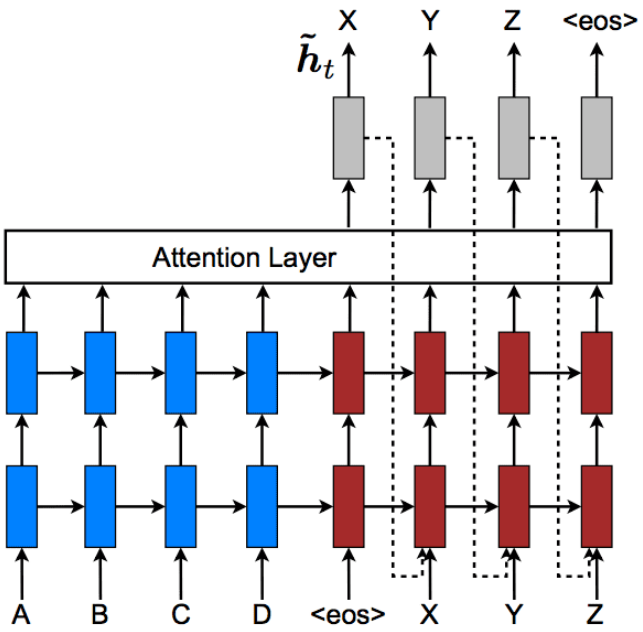

Figure 6: Attentional RNN encoder-decoder architecture (Luong et al., 2015)

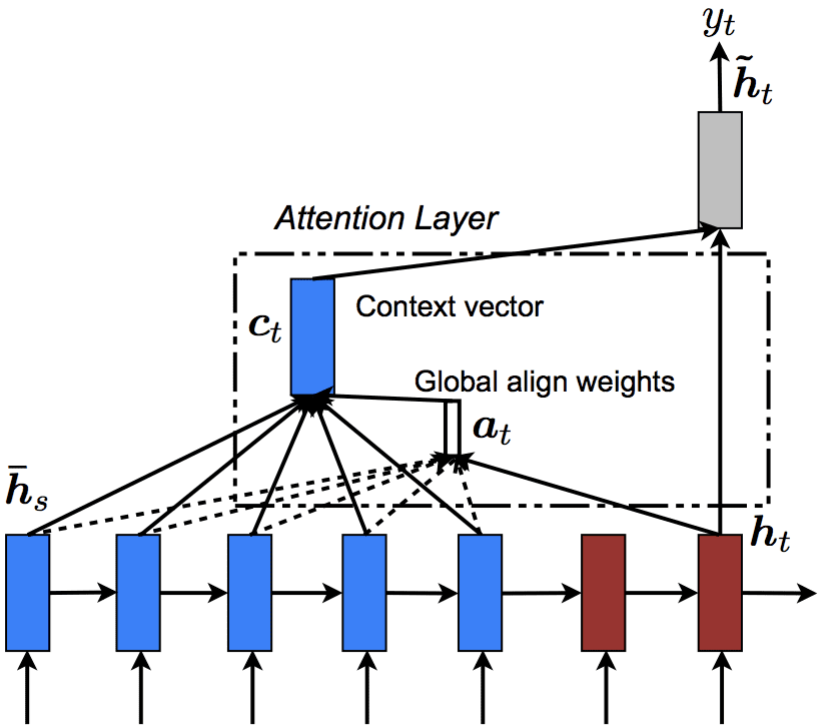

Figure 7: Depiction of how attention is calculated (Luong et al., 2015)

The **encoder** itself consists of a bidirectional GRU cell (Cho, van Merrienboer, Bahdanau, & Bengio, 2014) which takes a previous[3] hidden state h_{i-1} and an embedded input word tensor x_i and outputs an updated hidden state h_i and an output tensor y_i.

$$\text{GRU}_E(h_{i-1}, x_i) = h_i,\ y_i \tag{2}$$

[3]In the first time step, the hidden state is initialized to zeros.

The embedded word tensor has a dimension of 300, and because the cell is bidirectional, the output tensor and the hidden state are each composed of two 300 dimensional tensors (2×300), one for each direction. After processing an entire input sequence, the final hidden states are passed through a fully connected layer and transformed into a singular hidden state of size 300, while the outputs are simply concatenated in a tensor of size 600. The outputs of the encoder are then the final combined hidden state and the concatenated output tensors of the cell at each time step.

$$\text{Encoder}(x_{i...n}) = h_n,\ y_{i...n} \tag{3}$$

The **decoder** consists of an attention layer that calculates the context tensor c_i and a unidirectional GRU cell whose inputs are the hidden state h_{i-1} and the previously generated[4] input word w_i concatenated with the context tensor, which together have a dimension size of 900, meaning 300 each. Outputs of the cell are the updated hidden state h_i and the cell output tensor y_i, each of size 300 as well.

$$\text{GRU}_D(h_{i-1}, \text{cat}(w_{i-1}, c_i)) = h_i,\ y_i \tag{4}$$

Lastly, the cell output tensor, context tensor, and the embedded input tensor are passed through a fully connected layer to produce a word predictions tensor W_i which holds the probabilities for each word in the dictionary. This means that W_i has a dimension size equal to the size of the dictionary which is 33278.

$$\text{FC}(y_i, c_i, w_{i-1}) = W_i \tag{5}$$

$$\text{Decoder}(h_{i-1}, w_{i-1}, y_{i...n}) = W_i \tag{6}$$

Since the predicted word is used in the next time step of the decoder, W_i first needs to be converted to an index of the highest probability and then embedded using the embedding matrix.

$$w_i = \text{embedding\_matrix}[\text{argmax}(W_i)] \tag{7}$$

The model as described has 55M trainable parameters. While that describes most of the model, some of the details were omitted, such as the exact implementation of the attention layer and which activation functions were used. Luckily, all of that and more can be found on my github page.

3.4 Training the Model

The model was trained with a batch size of 16 across 15 epochs. The initial learning rate was set to $1 \cdot 10^{-3}$ and was gradually lowered using an exponential LR scheduler with a gamma of 0.85.

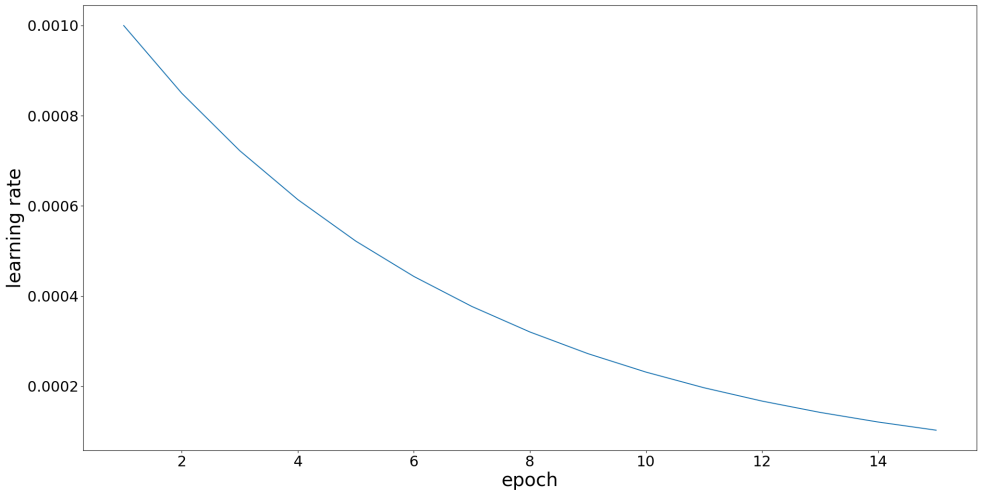

Figure 8: Learning rate through the epochs

During trainig, three key mechanisms were utilized: dropout, teacher forcing, and gradient clipping. Dropout layers were added as a form of regularization in both the encoder and the decoder acting on the embedded inputs; x_i in the encoder and w_i in the decoder. Both of the dropout layers were set to have a dropout probability of 0.5. Teacher forcing is a method that improves learning by sometimes feeding the decoder a word it *should* have generated, instead of the word it did generate. The rate of teacher forcing was set to 0.5. Lastly, gradient

[4] At the start of generation, a start-of-sentence token is given as an input word.

clipping is a must when training recurrent models since they often have issues with exploding gradients. To avoid this, gradient clipping is applied by calculating the norm of all gradients and then clipping the gradients above a certain set value, in this case it was set to 0.5.

Cross entropy was used as a loss function and its value was recorded every 20 batches:

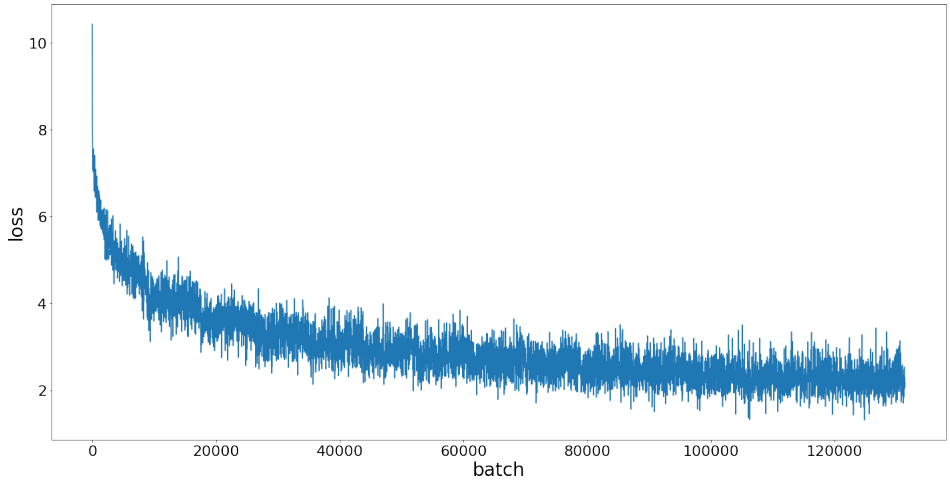

Figure 9: Loss during training

Model parameters were saved after each epoch so that early stopping could be used to detect overfitting, that is to decide which version of parameters to use when evaluating on the testing set. While the average training loss was the lowest in epoch 15, being approximately 2.2475, the average validation loss was lowest in epoch 14, amounting to 4.1270.

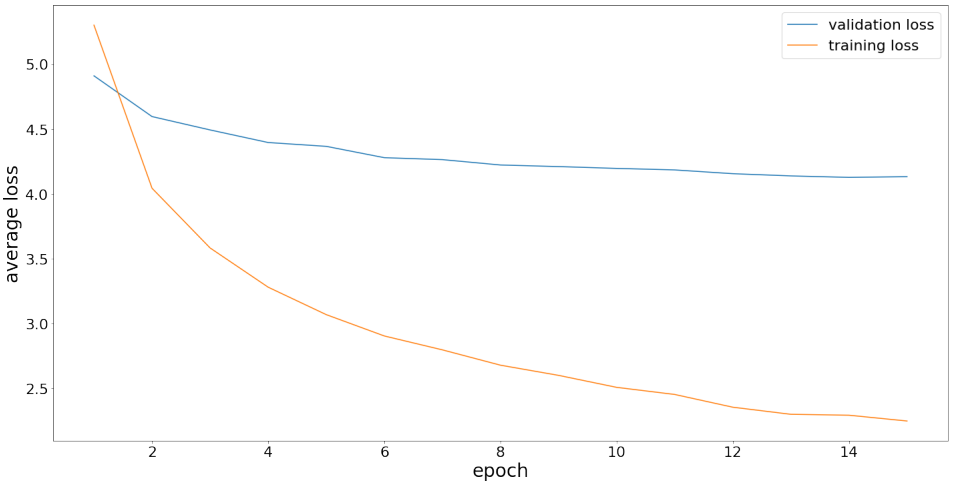

Figure 10: Average validation and training loss across epochs

Evaluating the model from epoch 14 on the testing set resulted in the average loss of 4.1252.

4 Results

To evaluate how well the model performs, we can utilize the ROUGE metric (Lin, 2004) which was originally designed to evaluate the quality of text summarizations. It compares the generated output with reference outputs and counts the overlapping units such as unigrams (R1), bigrams (R2) or subsequences (RL). Using ROUGE to compare generated titles against actual titles from the validation and test set yields the following results:

	R1	R2	RL
F-measure	48.78	25.06	46.62
Precision	53.58	27.50	51.09
Recall	47.58	24.74	45.58

Table 2: Evaluation of the model using the ROUGE metric

For comparison's sake, the following is the ROUGE table presented in the previously mentioned paper by Nallapati et al. (2016) which focuses on abstractive text summarization. Of course, title generation and text summarization are different tasks, but it helps to build intuition of the metric.

Model	Rouge-1	Rouge-2	Rouge-L
TOPIARY	25.12	6.46	20.12
ABS	26.55	7.06	22.05
ABS+	28.18	8.49	23.81
RAS-Elman	**28.97**	8.26	24.06
words-lvt2k-1sent	28.35	**9.46**	24.59
words-lvt5k-1sent	28.61	9.42	**25.24**

Table 3: ROUGE Recall table presented in Nallapati et al. (2016)

When it comes to individual examples of generated titles, the quality can varies from excellent to not so much, where the average lies somewhere in between. Here are some examples, T/G meaning True/Generated:

T/G	Title
T	testing the dark matter scenario for pev neutrinos observed in icecube
G	testing the dark matter with icecube pev neutrinos
T	phase transition in loop quantum gravity
G	phase transition in loop quantum gravity
T	simple alcohols with the lowest normal boiling point using topological indices
G	simple alcohols of the lowest normal boiling point
T	a note on list coloring powers of graphs
G	a note on the <UNK>graphs
T	complete solution of a gauged tensor model
G	complete analytic solution of the gauged gauged gauged gauged gauged gauged gauged gauged model
T	on various confidence intervals post model selection
G	on confidence intervals in the ¡UNK¿ model
T	collective motion of oscillatory walkers
G	collective motion of oscillatory walkers oscillatory walkers
T	outline of a generalization and a reinterpretation of quantum mechanics recovering objectivity
G	outline of a quantum model for quantum mechanics
T	variational quantum circuit model for knowledge graphs embedding
G	variational quantum circuit for knowledge graphs
T	robust information propagation through noisy neural circuits
G	robust information of information in limited limited neural networks
T	computer simulations of single particles in external electric fields
G	computer simulations of single particles in external electric fields
T	stability estimates for systems with small cross diffusion
G	stability estimates for nonlinear diffusion problems with nonlinear diffusion terms
T	malicious web domain identification using online credibility and performance data by considering the class imbalance issue
G	malicious web domain identification using deep learning
T	statistical tests for ¡UNK¿ pseudorandom number generator
G	statistical tests of pseudo random number pseudo ¡UNK¿ numbers
T	electrical properties of chain microstructure magnetic emulsions in magnetic field
G	electrical properties of chain microstructure magnetic emulsions
T	bringing quantum mechanics to life from schrödinger s cat to schrödinger s microbe
G	bringing quantum cat and the microbe
T	astrophysical neutrinos flavored with beyond the standard model physics
G	astrophysical neutrinos beyond astrophysical standard model

T/G	Title
T	dark energy from the gravity vacuum
G	dark energy in quantum gravity
T	quantum computation toolbox for decoherence free qubits using multi band alkali atoms
G	quantum engineering of qubits with ultracold atoms
T	probing a gravitational cat state
G	probing the cat state in a gravitational cat state
T	evolutionary techniques in lattice sieving algorithms
G	evolutionary lattice lattice lattice lattice lattice lattice
T	quantum dynamics of the early universe
G	quantum corrections to the early universe
T	non stationary spectral kernels
G	non stationary spectral kernels for non stationary kernel process
T	toward automatic verification of quantum programs
G	toward automatic verification of quantum programs
T	fifty years of energy extraction from rotating black hole revisiting magnetic penrose process
G	fifty years of a penrose process
T	transition probability estimates for long range random walks
G	transition and discrete step rates for discrete time spaces
T	qubit entanglement across epsilon near zero media
G	qubit materials near near materials materials materials
T	equations defining probability tree models
G	models of the trees in tree tree
T	graphene plasmonics
G	graphene plasmonics graphene plasmonics
T	a refined analysis on the x 3872 resonance
G	a study d bar d j d j d j d j d j d d d j d
T	asymptotic equivalence of quantum stochastic models
G	asymptotic convergence of quantum stochastic models
T	energy level statistics in strongly disordered systems with power law hopping
G	energy level statistics statistics of disordered disordered systems
T	on the various aspects of electromagnetic potentials in spacetimes with symmetries
G	on the killing killing killing killing killing killing killing killing killing killing killing killing killing vector fields

Table 4: Examples of generated sentences

It is also clear that model suffers from neural text degeneration meaning it repeats some words or phrases multiple times. It is an interesting phenomenon and there are proposed solutions to combat it (Holtzman, Buys, Du, Forbes, & Choi, 2019). All of the generated titles, alongside the model itself and the dataset can be found on my github page.

5 Discussion

There are many factors which influence how any given model will perform. In this particular case, there are several ways in which the model could be improved. One such way would be a better sanitation and tokenization of the dataset. For example, numbers and similar symbols could be removed from the vocabulary and a previously mentioned switching generator-pointer mechanism (Nallapati et al., 2016) could be implemented to deal with words not present in the dictionary. Of course, a better dataset could be used as well, one less scientific, perhaps.

One potentially simple improvement would be to implement a separate embedding matrix for the decoder which would increase the number of trainable parameters significantly, but as GPT-3 has shown, when it comes to language models, bigger is better.

As mentioned, the model suffers from a case of neural text degeneration. An interesting experiment would be to implement Nucleus Sampling proposed by Holtzman et al. (2019) to see if the problem persists.

It is also possible a slightly different model configuration could have yielded better results since the single greatest obstacle in the way of achieving such results in the scope of this paper was certainly the hardware available for training. The model was implemented to be highly customizable and as such it could have been easily modified and tested with many different configurations, if the hardware were to allow it. Which sadly wasn't the case here. With more and better GPUs, batch size could be increased, training time reduced resulting in an easier time of fine-tuning the hyperparameters for this specific task. If none of that yields satisfactory results, the next logical step would be to move onto the transformer architecture.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural machine translation by jointly learning to align and translate.* arXiv. Retrieved from https://arxiv.org/abs/1409.0473 DOI: 10.48550/ARXIV.1409.0473

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). *Language models are few-shot learners.* arXiv. Retrieved from https://arxiv.org/abs/2005.14165 DOI: 10.48550/ARXIV.2005.14165

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). *On the properties of neural machine translation: Encoder-decoder approaches.* arXiv. Retrieved from https://arxiv.org/abs/1409.1259 DOI: 10.48550/ARXIV.1409.1259

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014, October). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1724–1734). Doha, Qatar: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D14-1179 DOI: 10.3115/v1/D14-1179

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *Bert: Pre-training of deep bidirectional transformers for language understanding.* arXiv. Retrieved from https://arxiv.org/abs/1810.04805 DOI: 10.48550/ARXIV.1810.04805

Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2019). *The curious case of neural text degeneration.* arXiv. Retrieved from https://arxiv.org/abs/1904.09751 DOI: 10.48550/ARXIV.1904.09751

Karpathy, A. (2015). *The unreasonable effectiveness of recurrent neural networks.* Retrieved 2022-12-09, from https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... Zettlemoyer, L. (2019). *Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.* arXiv. Retrieved from https://arxiv.org/abs/1910.13461 DOI: 10.48550/ARXIV.1910.13461

Li, P., Lam, W., Bing, L., & Wang, Z. (2017). *Deep recurrent generative decoder for abstractive text summarization.* arXiv. Retrieved from https://arxiv.org/abs/1708.00625 DOI: 10.48550/ARXIV.1708.00625

Lin, C.-Y. (2004, July). ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74–81). Barcelona, Spain: Association for Computational Linguistics. Retrieved from https://aclanthology.org/W04-1013

Luong, T., Pham, H., & Manning, C. D. (2015, September). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1412–1421). Lisbon, Portugal: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D15-1166 DOI: 10.18653/v1/D15-1166

Nallapati, R., Zhou, B., santos, C. N. d., Gulcehre, C., & Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. Retrieved from https://arxiv.org/abs/1602.06023 DOI: 10.48550/ARXIV.1602.06023

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). *Attention is all you need.* arXiv. Retrieved from https://arxiv.org/abs/1706.03762 DOI: 10.48550/ARXIV.1706.03762

Yadav, D., Desai, J., & Yadav, A. K. (2022, 03). *Automatic text summarization methods: A comprehensive review.*

Yang, M., Li, C., Shen, Y., Wu, Q., Zhao, Z., & Chen, X. (2021). Hierarchical human-like deep neural networks for abstractive text summarization. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(6), 2744-2757. DOI: 10.1109/TNNLS.2020.3008037

Zhang, Y., Li, D., Wang, Y., Fang, Y., & Xiao, W. (2019, 04). Abstract text summarization with a convolutional seq2seq model. *Applied Sciences*, *9*, 1665. DOI: 10.3390/app9081665