Text Summarization Using Deep Neural Networks

Filippa Kärrfelt karrfe@kth.se
Isak Pettersson isakpet@kth.se
Mikael Törnwall tornwall@kth.se

May 25, 2021

Abstract

In this project different techniques for text summarization using deep neural networks are explored. The work includes methods for both abstractive and extractive summarization. The methods were applied on two different datasets; one with shorter and fewer documents and one with longer and a higher number of documents. To guide the exploration, the extractive summarization method was based on an existing research project. In order to compare results of different methods and to assess the quality of the produced summaries the predefined ROUGE scoring scheme was utilized. There were significant differences in performance between the datasets, which could potentially be explained by their different compositions. The results indicate that the extractive summarizer generally performed better on the smaller dataset and the abstractive summarizer performed better on the larger dataset. Generally, the ROUGE scores obtained were significantly lower that the scores presented in the reference paper.

1 Introduction

The aim of text summarization is to create a condensed version of an original text. This can be done manually by human experts, but is then an expensive task in terms of time and resources. The need to summarize text is not new and therefore research on automatic techniques data back to as early as 1958 [1]. Within text summarization there are two main techniques; abstractive and extractive summarization [1]. In abstractive summarization the model is generating summaries using words in the corpus, while in extractive summarization the model is selecting sentences in the corpus that will then be considered the summary. Within this project both techniques are explored using variations of deep autoencoders (DAE).

For the extractive summarization task a technique using a DAE was used to generate the summaries. This exploration was following the methodology of Yousefi-Azar et al. [1] for determining the DAE architecture and techniques for pre-training the weights using a layered Restricted Boltzmann Machine (RBM). The sentence selection when creating the summaries use the TextRank method described by Prateek [2]. Furthermore, the reference paper was extended to also use sentence representation based on GloVe word vectors [2].

For the abstractive text summarization task a technique with a sequence to sequence network [3] with an autoencoder formed from two recurrent neural networks combined with an attention mechanism was used [4].

Both summarization techniques were tested on two datasets. The BC3 annotated email dataset [5] and a dataset consisting of podcast transcripts provided by Spotify [6]. The generated summaries were evaluated against the reference summaries using the ROUGE metric [7].

The implementation is publicly accessible at `https://github.com/MikaelTornwall/dd2424_project`. We used the PyTorch library and ran the heaviest computing tasks on Google Cloud Platform.

2 Related work

Deep neural networks have been used for both abstractive and extractive summarization. For extractive summarization Yousefi-Azar et al. [1] introduced a technique using a deep autoencoder. In their method they utilize stacked RBMs to find good parameter values to use when initializing the autoencoder, as model initialization has shown great importance for performance of the final model [8]. Much of the novelty introduced by Yousefi-Azar et al. stem from their sentence vector representations, in which there has been a rapid development since their work was published. Thus, for more recent projects on extractive text summarization pre-trained GloVe word representations [9] have been utilized [2] [10]. Verma and Nidhi also explored using a method similar to Yousefi-Azar et al. but with further enhancements to the sentence vector features [11].

For abstractive summarization we used an implementation of a sequence to sequence (Seq2Seq) network [4]. Seq2Seq networks have been successful in neural machine translation tasks as showed by Sutskever et al. [3] and Bahdanau et al. [12]. Other application areas include for example conversational models, as proposed by Vinyals and Le [13], and text summarization tasks [14]. We found this to be the most suitable approach for abstractive summarization.

3 Data

3.1 BC3 emails

For building and testing our implementations we used BC3 email corpus [5] consisting of 259 unique emails, each containing one or more alternative summaries. This email corpus was also used in the reference paper [1]. We further filtered this set down to a subset of 127 email body-summary pairs, s.t. each email body had a minimum length of 50 words.

3.2 Spotify podcast transcripts

To extend our experiments, we used a dataset from Spotify [6] that consisted of a number of podcast transcriptions with corresponding descriptions. For our experiments we are using the descriptions as a summary, but those are of varying quality between the podcast episodes. Since the bulk of the data are transcripts of spoken language, also those are of varying length and quality. We extracted a subset of 422 transcription-description pairs, where the maximum length of the transcription was 10 000 words and the maximum length of a description was 100 words. An additional test set containing 45 transcription-description pairs was also created.

3.3 GloVe word vectors

To create sentence vector representations pre-trained GloVe word vectors [9] were utilized.

4 Methods

4.1 Data parsing

Each of the datasets are parsed from their original format to pandas DataFrame format. In addition to the email body-summary and transcription-description pairs, the DataFrames contain additional information, such as titles. The main text body is in string-format that may require more parsing depending on the summarization tool. An additional list of tokenized sentences is also saved in the DataFrame. This contains each sentence separated, translated into lowercase and trimmed from stopwords and punctuation.

For the seq2seq model [4] each text is stripped-down to include only characters a-z in lower case. Also all punctuation, stopwords, URL and email addresses are removed.

4.2 Input vector representations

In order to apply the deep learning techniques to the text data, it needs to be translated into suitable vector representations. In the project, the trimmed sentences in the original datasets are translated into

their vector representations. For the extractive summarization technique two types of sentence vector representations have been used and compared. The first sentence representation is the document wise tf-idf described by Yousefi-Azar et al. [1] and the other is the GloVe sentence vector representation described by Prateek [2].

For the abstractive summarization approach, one-hot encoded word vector representations are created for each word in a language. A language is just the total set of unique words in all the training data. Thus, the corresponding one-hot encoded word vector is a vector by the length of the unique words in a language. Additionally, each word has a unique index in the language, so the index determines which element in the vector representation is one while the rest being zeros. In practice these vectors are represented as tensors containing the indices of each word [4].

4.3 Evaluation metric

For evaluating our summaries we used the Recall-Oriented Understudy for Gisting Evaluation (ROGUE) metrics.[7] ROGUE is a set of metrics for assessing the performance of automatic summaries or translations. It evaluates by comparing automatically produced summaries against a set of reference summaries (usually created by humans).

For our tests we focused on the ROGUE-1 and ROGUE-2 metric-scores to assess our results. Both these metrics come with three different scores for evaluation. Precision, Recall and F-score. Precision is a measure of how much of the system summary was in fact relevant or needed [15]. Recall refers to how much of the reference summary the system summary is recovering or capturing [15]. F-score is a measure for the accuracy of a summary and is calculated using the precision and recall. To compute the scores for the created summaries we used the ROGUE python package.[16]

4.4 Extractive summarization using deep autoencoder

To create the extractive summarizer the methodology described by Yousefi-Azar et al. [1] was used. This included creating the tf-idf vector representations, the DAE architecture and the method for pre-training the model parameters using stacked RBMs. As this paper did not come with any code reference, we used an additional source for inspiration in creating the RBM and DAE using PyTorch [17].

4.4.1 Stacked RBM for coarse parameter search

As described in the reference paper [1] we performed a course search in order to find good initial parameter values for the DAE by using stacked RBMs. We also used the same hidden layer dimensions (140, 40, 30, 10) for our model. When setting up the RBM architecture we took inspiration from several articles regarding how to build stacked RBMs such as [18], [19] and [17]. For the final model we decided for an implementation very similar to [17] that is largely based on the paper [8]. A paper that is also referenced in the original paper that we have attempted to replicate.

Similarly to the reference paper this implementation uses contrastive divergence for updating the parameters of individual RBMs, updating based on the difference between the input data and the data reconstructed by the RBM. By using the current RBM we then calculate new data that is used as input for the RBM in the next layer, and by doing this we are able to stack them. During training all the intermediate weights are saved in order to be used later during the fine tuning phase.

4.4.2 Fine tuning, applying back- propagation

In the fine tuning phase the weights of every layer previously calculated by the stacked RBMs are used for weight initialization of the DAE when back-propagating. For fine tuning we tested using both the cross-entropy error as our loss-function just like the original paper as well as the mean square error.

For optimization of the parameters we deviated a bit from the original paper and used an Adam optimization instead of a mini-batch Conjugate Gradient with line search and Polak-Ribiere rule. We initially tried to use the library `https://pypi.org/project/CGDs/` for implementing a Conjugate Gradient optimizer but were not successful in integrating this with our project. Thus we opted to use a different optimizer instead.

4.4.3 Sentence selection using TextRank

The reference paper sentences to include in the summary were selected based on the cosine similarity to the email subject [1]. In this approach it was then assumed that sentences similar to the email subject would be relevant for the summary. In this part of the project we deviated from the original study, and instead ranked sentences based on the TextRank algorithm.

The TextRank algorithm is inspired by the PageRank algorithm [20], but instead of using web pages TextRank is using sentences and instead of transition probabilities it is using cosine similarity between vector representations of sentences [2]. By applying the TextRank algorithm to a set of sentence vectors for a document, we can extract the top ranked sentences and use those to create the summary.

4.5 Abstractive summarization using sequence to sequence network

As an alternative for the extractive approach to summarization, we implemented two sequence to sequence networks [3], in which two recurrent neural networks form an autoencoder. To enhance the autoencoder's ability to recognize important features in the data, an attention mechanism was included in the second decoder implementation. One-hot encoded vectors were used as the input. Our implementation was inspired by PyTorch NLP translation Seq2Seq implementation [4], but applied to a text summarization task.

Architecture-wise we ended up using Gated Recurrent Unit (GRU) in the encoder and both decoder implementations. We decided to use GRUs instead of LSTMs as GRUs potentially performs better with smaller data sets and are computationally more efficient, which were both preferred qualities considering our experiments [21]. In addition, the attention decoder was constructed using two linear layers for the attention mechanism and a linear output layer, as in the reference implementation [4]. For more details, see Appendix D.2.

5 Experiments

5.1 Training and model validation

In both the abstractive and extractive summarization methods we train the models to obtain the final settings. To validate that the training is actually improving the models, we computed and plotted the loss throughout the training process.

For the extractive method we used two different loss functions; the cross-entropy loss (CELoss) when training on the tf-idf vectors and the mean-square error (MSE) when training on the GloVe vectors. In the reference paper the CELoss is used [1], but using CELoss is not suitable for the GloVe vectors. Other than verifying that the loss is decreasing as training progresses, we wanted to validate that initializing the DAE with the model parameters provided by the RBMs did in fact decrease the initial loss. In order to verify this, we made it possible to initialize the DAE without the RBM trained parameters. When making the comparison between initializations we concluded that the loss is in fact decreasing, and by initializing the DAE with the RBM parameters the initial loss is significantly lower. For the BC3 dataset we are also able to achieve a smaller final loss when using RBM initialization.

For the Seq2Seq model the negative log likelihood loss (NLLLoss) was used as in the referenced source [4]. When training on the Spotify data the loss starts to decline steadily after first couple thousand iterations. The loss starts to converge only after 10k iterations. Compared to the smaller BC3 dataset it takes several thousand iterations more. This is likely due to the size difference between the two datasets. See the detailed results and plots in Appendix A.1.

5.2 Evaluation

Several different versions of the methods were tested and evaluated. In the tables below we display the ROGUE-scores for these different runs. 'Corpus' refers to the model being trained on the whole set of sentences in the input while 'Document' means that the model was trained on individual documents separately. The two top sections include the results produced by the extractive summarizer while the last show the scores for the abstractive. The two sections for the extractive summarizer results correspond to the results for the different sentence vector representations (tf-idf or GloVe).

5.2.1 BC3-dataset

Table 1: ROUGE-1 scores for the BC3 email corpus

Model	Precision	Recall	F-score
tf-idf	0.22	0.243	0.2
Corpus DAE (tf-idf)	0.172	0.142	0.153
Corpus DAE + RBM (tf-idf)	0.135	0.128	0.12
Document DAE + RBM (tf-idf)	0.164	0.148	0.124
GloVe	0.117	0.229	0.143
Corpus DAE (GloVe)	0.122	0.124	0.117
Corpus DAE + RBM (GloVe)	0.167	0.184	0.166
Document DAE + RBM (GloVe)	0.148	0.164	0.128
Seq2Seq (without attention)	0.099	0.087	0.083
Seq2Seq (with attention)	0.114	0.122	0.115

The models are trained on the BC3 email corpus containing 127 emails with each containing more that 50 words, and the ROUGE-1 scores are computed using the abstractive summaries created by humans as a reference.

Looking at the ROUGE-1 results, extractive approach seems to outperform the abstractive approach. The same trend does not continue over to ROUGE-2 metric (see Appendix B.1), although overall the extractive approach still produces the best results. Now however, several extractive approaches are outperformed by the Seq2Seq model with attention. As expected Seq2Seq with attention outperforms Seq2Seq without attention mechanism across each individual metric. When comparing the different extractive approaches for ROUGE-1 we can see that the overall highest score is produced by the raw tf-idf representation.

5.2.2 Spotify-dataset

Table 2: ROUGE-1 scores for the BC3 email corpus

Model	Precision	Recall	F-score
tf-idf	0.066	0.084	0.061
Corpus DAE (tf-idf)	0.059	0.137	0.075
Corpus DAE + RBM (tf-idf)	0.055	0.156	0.077
Document DAE + RBM (tf-idf)	0.046	0.037	0.04
GloVe	0.049	0.127	0.064
Corpus DAE (GloVe)	0.043	0.106	0.055
Corpus DAE + RBM (GloVe)	0.04	0.063	0.045
Document DAE + RBM (GloVe)	0.044	0.062	0.048
Seq2Seq (without attention)	0.225	0.183	0.202
Seq2Seq (with attention)	0.210	0.171	0.187

The models are trained on the selected subset of Spotify transcription corpus containing 422 training samples, and the ROUGE-1 scores are computed using the abstractive episode description.

Looking at the results, it seems that the Seq2Seq model benefited from the increase in the size of the data samples and the number of data samples that was gained from switching datasets from BC3 to the Spotify dataset. The opposite happened to the extractive models, as both ROUGE-1 and ROUGE-2 performance dropped significantly (see Appendix B.1 for ROUGE-2 scores). This is likely due to difference in how extractive and abstractive approaches build the output sentences; extractive models pick the most probable sentence from the transcripts and use that as the output sentence. As the transcripts are fairly long compared to the email bodies, the probability that a single sentence from the transcripts could encompass much of the information contained in the summary seems unlikely. The extractive summaries were kept to one sentence as the summaries produced by the abstractive

method were short, but we did an exploration on summary length which can be found in Appendix D.1.

On the other hand, abstractive models are more flexible in using the complete vocabulary of the language. After analyzing some of the output sentences it seems like the Seq2Seq model sometimes identifies which actual podcast is in question and reconstructs its output from the description of another episode of the given podcast. An unexpected result was that Seq2Seq network without attention mechanism outperformed the network with attention mechanism.

5.3 Descriptions

Below is an interesting example that two of our models produced paired with the target description from the Spotify dataset. For more summaries, see Appendix C.

Original reference: *Chris Ryan has a strong opinion on smoking*

Reference after removing stopwords: *chris ryan strong opinion smoking*

Autoencoder (with RBM), GloVe: *There's like two or three people who smoked cool in this world Jon Hamm Denis Leary and me*

Seq2Seq (with attention): *chris ryan gives perspective future historical netflix drama*

6 Conclusion

We found it quite interesting to be able to apply both the abstractive and the extractive model on two different datasets, and it is feasible to extend the methods to other datasets as well. During the exploration phase we have elaborated with different model settings and architectures, but in that area there is much more to be done. For the abstractive summarizer we have made some investigation regarding the optimal number of iterations to train for and the optimal number of nodes in the hidden layers, but as the training times are long, this could be explored further. Another area to explore is using a different number of layers and different types of layers, such as LSTMs, if we'd have more data, which we believe could generate better models.

For the extractive summarizer the final RBM and autoencoder architecture is mimicking the architecture of the reference paper [1], but was not actually verified to be the optimal architecture. As we deviated from the reference paper by using GloVe vectors, we also did some trials with different loss functions for the DAE training but we are unsure of the effects of these differences.

Generally, we found it quite challenging to follow the reference paper, as it in some sections gave very little explanation on the technicalities of the applied methods. Thus, we are unsure if the details in the implementations are actually the same. In the reference paper it is mentioned that they use mini-batch Conjugate Gradient descent with line search and the Polak-Ribiere rule for search direction as the optimizing function for the DAE, but we were not able to implement this. Instead we use the Adam optimizer, but are unsure about the effect of this deviation.

We have developed and explored both abstractive and extractive methods using deep learning for text summarization. The specific deep learning methods we have used includes Recurrent Neural Networks, Deep autoencoders and Restricted Boltzmann machines. We have deviated slightly from our project proposal by applying changes to the methodology in the reference paper, and by extending the project to also include an abstractive summarization method. The ROUGE scores that we were able to achieve with our models are worse than the scores presented in the original paper, but this was expected as we did not completely follow the original implementation, and had only so much time to search for optimal architectures and parameter settings. For the abstractive methods we did not have any initial reference values, as the referenced implementation [4] was applied in a different context. However, the extractive summaries and the paper by Yousefi-Azar et al [1] provided decent reference values.

References

[1] M. Yousefi-Azar and L. Hamey, "Text summarization using unsupervised deep learning," *Expert Systems with Applications*, vol. 68, pp. 93–105, 2017, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2016.10.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417416305486.

[2] J. Prateek, *An introduction to text summarization using the textrank algorithm (with python implementation)*, 2018. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/?utm_campaign=News&utm_medium=Community&utm_source=DataCamp.com (visited on 05/18/2021).

[3] I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to sequence learning with neural networks*, 2014. arXiv: 1409.3215 [cs.CL].

[4] S. Robertson, *Nlp from scratch: Translation with a sequence to sequence network and attention*. [Online]. Available: https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html (visited on 05/21/2021).

[5] J. Ulrich, G. Murray, and G. Carenini, "A publicly available annotated corpus for supervised email summarization," in *AAAI08 EMAIL Workshop*, Chicago, USA: AAAI, 2008. [Online]. Available: https://www.cs.ubc.ca/cs-research/lci/research-groups/natural-language-processing/bc3.html.

[6] A. Clifton, S. Reddy, Y. Yu, A. Pappu, R. Rezapour, H. Bonab, M. Eskevich, G. Jones, J. Karlgren, B. Carterette, and R. Jones, "100,000 podcasts: A spoken English document corpus," in *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 5903–5917. [Online]. Available: https://www.aclweb.org/%20anthology/2020.coling-main.519.

[7] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://www.aclweb.org/anthology/W04-1013.

[8] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006, ISSN: 0036-8075. DOI: 10.1126/science.1127647. eprint: https://science.sciencemag.org/content/313/5786/504.full.pdf. [Online]. Available: https://science.sciencemag.org/content/313/5786/504.

[9] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162.

[10] Kirt? *Enron email text summarization*, 2019. [Online]. Available: https://github.com/dailykirt/ML_Enron_email_summary (visited on 05/18/2021).

[11] S. Verma and V. Nidhi, *Extractive summarization using deep learning*, 2019. arXiv: 1708.04439. [Online]. Available: https://arxiv.org/abs/1708.04439.

[12] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2016. arXiv: 1409.0473 [cs.CL].

[13] O. Vinyals and Q. Le, *A neural conversational model*, 2015. arXiv: 1506.05869 [cs.CL].

[14] A. Pai, *Comprehensive guide to text summarization using deep learning in python*, 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/ (visited on 05/23/2021).

[15] K. Ganesan, *An intro to rouge, and how to use it to evaluate summaries*, 2017. [Online]. Available: https://www.freecodecamp.org/news/what-is-rouge-and-how-it-works-for-evaluation-of-summaries-e059fb8ac840/ (visited on 05/18/2021).

[16] *Python library for rouge metric*, 2020. [Online]. Available: https://pypi.org/project/rouge/ (visited on 05/18/2021).

[17] E. Tang, *Improving autoencoder performance with pretrained rbms*, 2020. [Online]. Available: https://towardsdatascience.com/improving-autoencoder-performance-with-pretrained-rbms-e2e13113c782 (visited on 12/17/2020).

[18] V. Kurama, *Beginner's guide to boltzmann machines in pytorch*, 2021. [Online]. Available: https://blog.paperspace.com/beginners-guide-to-boltzmann-machines-pytorch/ (visited on 05/18/2021).

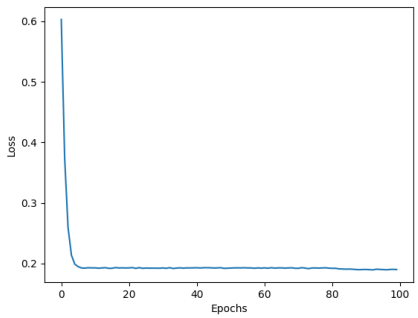
[19] D. Mwiti, *Introduction to restricted boltzmann machines using pytorch*, 2018. [Online]. Available: https://heartbeat.fritz.ai/guide-to-restricted-boltzmann-machines-using-pytorch-ee50d1ed21a8 (visited on 05/18/2021).

[20] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks*, vol. 30, pp. 107–117, 1998. [Online]. Available: http://www-db.stanford.edu/~backrub/google.html.

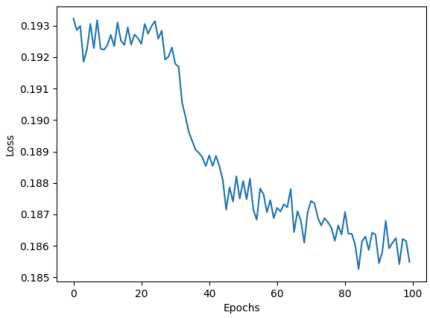
[21] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, 2014. arXiv: 1412.3555 [cs.NE].

A Appendix

A.1 BC3-dataset loss plots

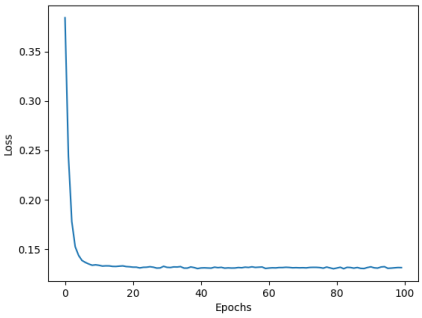


(a) DAE tf-idf

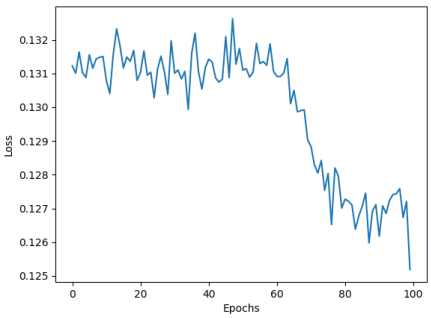


(b) DAE + RBM tf-idf

Figure 1: Cross-entropy loss plots for the tf-idf vectors with and without RBM initialization

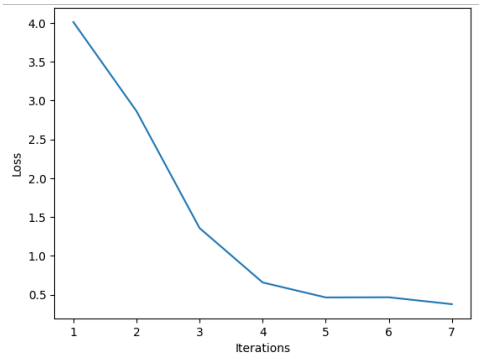


(a) DAE GloVe

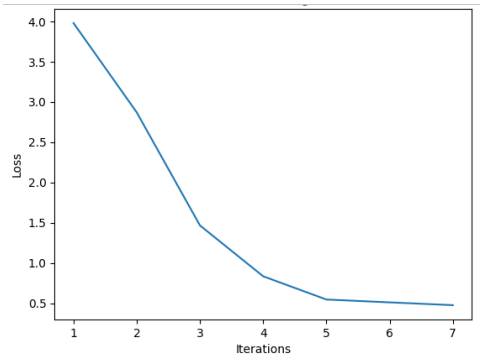


(b) DAE + RBM GloVe

Figure 2: MSE loss plots for the GloVe vectors with and without RBM initialization



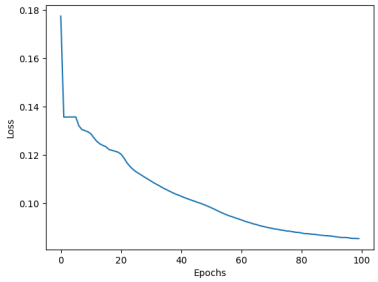
(a) Seq2Seq NLLLoss



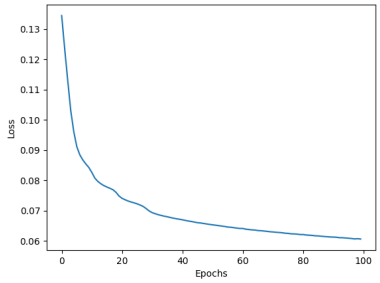
(b) Seq2Seq + attention NLLLoss

Figure 3: (a) Loss plot for Seq2Seq model without attention, 7000 iterations on the BC3 dataset. (b) Loss plot for Seq2Seq model with attention, 7000 iterations on the BC3 dataset.

A.2 Spotify dataset loss plots

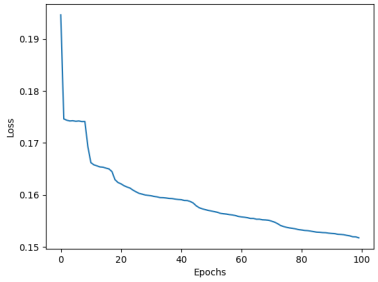


(a) DAE tf-idf

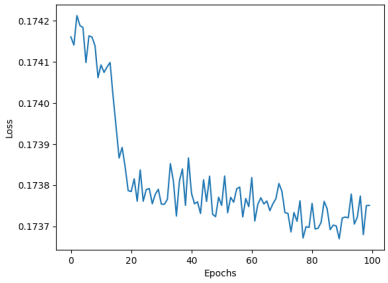


(b) DAE + RBM tf-idf

Figure 4: Cross-entropy error loss plots for the tf-idf vectors with and without RBM initialization. Ran for 100 times with 100 epochs in every iteration.

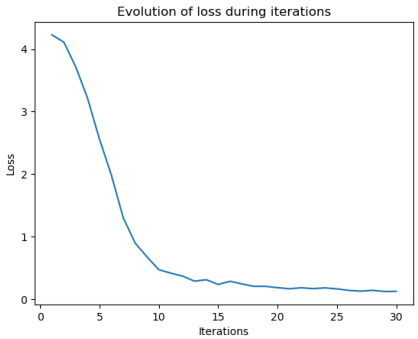


(a) DAE GloVe

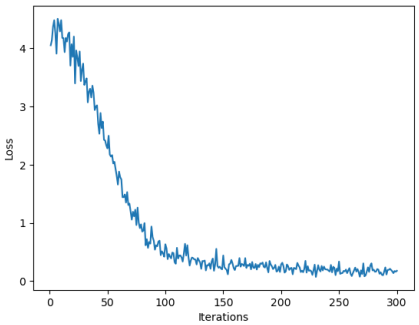


(b) DAE + RBM GloVe

Figure 5: MSE loss plots for the GloVe vectors with and without RBM initialization. Ran for 100 times with 100 epochs every iteration.



(a) Seq2Seq



(b) Seq2Seq + attention

Figure 6: Negative Log Likelihood loss plot for Seq2Seq network without and with attention, 30 000 iterations on the Spotify dataset *Note: the volatility if figure (b) is due to the more consistent data points*

B Appendix

B.1 ROUGE-2 scores for BC3 and Spotify datasets

Table 3: ROUGE-2 scores for the BC3 email corpus

Model	Precision	Recall	F-score
tf-idf	0.08	0.082	0.072
Corpus DAE (tf-idf)	0.062	0.047	0.053
Corpus DAE + RBM (tf-idf)	0.066	0.054	0.059
Document DAE + RBM (tf-idf)	0.015	0.029	0.019
GloVe	0.024	0.06	0.032
Corpus DAE (GloVe)	0.021	0.017	0.018
Corpus DAE + RBM (GloVe)	0.072	0.058	0.064
Document DAE + RBM (GloVe)	0.015	0.021	0.014
Seq2Seq (without attention)	0.003	0.005	0.004
Seq2Seq (with attention)	0.067	0.057	0.062

The models are trained on the BC3 email corpus containing 127 emails with each containing more that 50 words, and the ROUGE-2 scores are computed using the abstractive summaries created by humans as a reference.

Table 4: ROUGE-2 scores for the Spotify dataset

Model	Precision	Recall	F-score
tf-idf	0.014	0.009	0.009
Corpus DAE (tf-idf)	0.01	0.028	0.014
Corpus DAE + RBM (tf-idf)	0.011	0.028	0.015
Document DAE + RBM (tf-idf)	0.007	0.005	0.006
GloVe	0.005	0.006	0.005
Corpus DAE (GloVe)	0.001	0.003	0.001
Corpus DAE + RBM (GloVe)	0.001	0.004	0.002
Document DAE + RBM (GloVe)	0.016	0.046	0.013
Seq2Seq (without attention)	0.200	0.148	0.170
Seq2Seq (with attention)	0.150	0.119	0.132

The models are trained on the selected subset of Spotify transcription corpus containing 422 training samples, and the ROUGE-2 scores are computed using the abstractive episode description.

C Appendix

C.1 Summaries

Original reference: *we talk about boomers*

Reference after removing stopwords: *talk boomers*

Autoencoder (with RBM), GloVe: *What does that even mean?*

Seq2Seq (with attention): *boy three heads enters pie eating contest*

Original reference: *You and I are made in the Likeness of God. We were created to manifest the glory of God.*

Reference after removing stopwords: *made likeness god created manifest glory god*

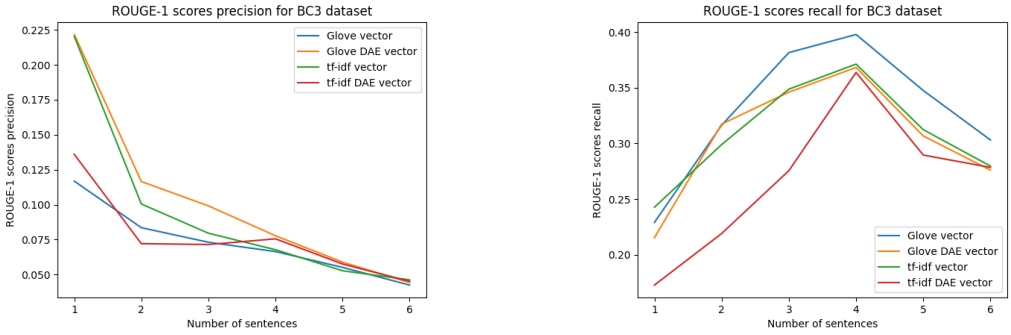
Autoencoder (with RBM), GloVe: *up to 1 verse 27 God created man in His image and likeness male and female he created them and it's the most beautiful thing to actually realize that you are made in an image and likeness of God there in you that is God that people should see God in you that people should glorify the god that lives in you and what is the most powerful thing than to appreciate that?*

Seq2Seq (with attention): *god prepares us something greater*

D Appendix

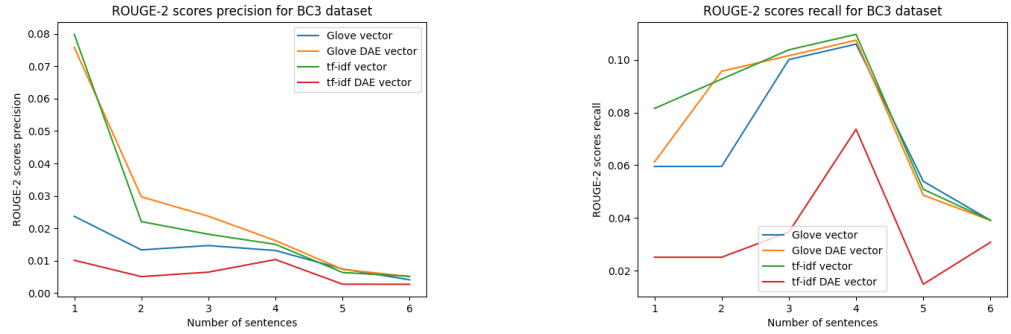
D.1 Exploration on summary length for the extractive summarizer

For the BC3 dataset the recall seem to be at an optimal value when using 4 sentences in the summary, but the reason for this is likely that at that point a majority of the sentences that exist in each document are included in the summary.



(a) precision (b) recall

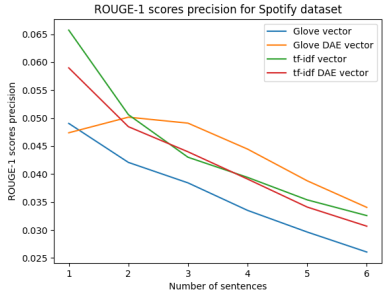
Figure 7: Rouge-1 precision and recall scores for 1-6 sentence summaries produced by the extractive summarizer for the BC3 dataset



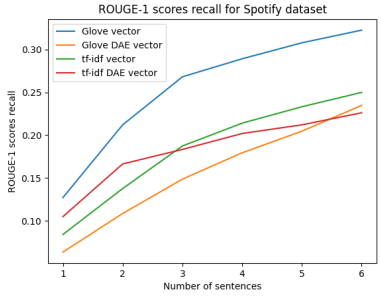
(a) precision (b) recall

Figure 8: Rouge-2 precision and recall scores for different number of sentences produced by the extractive summarizer for the BC3 dataset

The extractive summary is created by combining the by TextRank n highest ranked sentences. In figure 4 the ROUGE-1 scores for precision and recall for different models and summary lengths for the Spotify dataset are presented. The same comparisons for the BC3 dataset are available in the appendix section A.1.



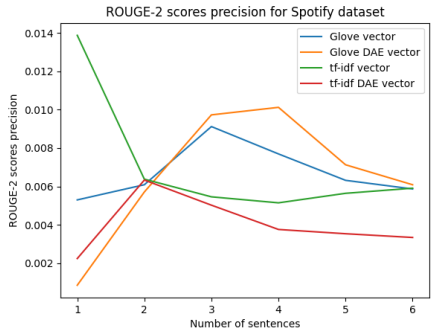
(a) precision



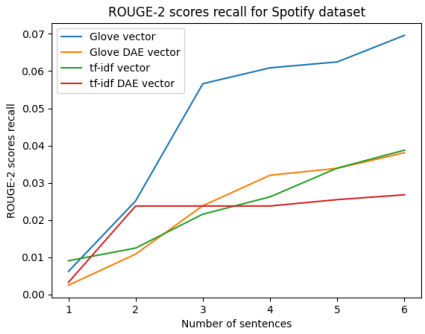
(b) recall

Figure 9: Rouge-1 precision and recall scores for 1-6 sentence summaries produced by the extractive summarizer for the Spotify dataset

In figure 4 we can see the average ROUGE-1 scores for the raw tf-idf and GloVe vectors, and the representations produced by models trained using the RBM + DAE. The results are quite ambiguous, but over several runs it seems as though the vectors produced by the DAE outperform the raw vectors on precision, and the raw vectors outperform the ones produced by the DAE on recall.



(a) precision



(b) recall

Figure 10: Rouge-2 precision and recall scores for different number of sentences produced by the extractive summarizer for the Spotify dataset

D.2 A search for an optimal size for the hidden layer in Seq2Seq network

A search for an optimal size for the hidden layer was conducted between 128, 256 and 512 nodes. We ended up using hidden layer size of 256, as it yielded the highest ROUGE-1 scores across each metric; precision, recall and F-score, when attention was included.

Table 5: Search for optimal hidden layer size, ROUGE-1 scores, Seq2Seq with and without attention

Hidden nodes	Precision (attn)	Recall (attn)	F-score (attn)	Precision	Recall	F-score
128	0.048	0.089	0.057	0.019	0.030	0.023
256	0.061	0.095	0.070	0.014	0.036	0.019
512	0.054	0.082	0.053	0.027	0.050	0.034

1000 iterations on the BC3 dataset