ICNS
B729339

All you need is focus: A novel method for abstractive based text summarisation technology

by

Alexander Collie

A Project Report

Submitted in partial fulfilment
of the requirements for the award of

B.Sc.
in
Computer Science
of
Loughborough University

4th May 2022

Abstract

In this project: Current language transformers are limited by only including recently seen events in their summaries, we refer to this as having narrow attention and too much 'focus'. This project proposes 'bucketing', a novel method of abstractive-based summarisation to reset the attention and thus generalise the focus.

Practical use: To demonstrate the bucketing system, a website created using Django and Vue.js, allows users to search for events and summarize news articles. The site automatically fetches the latest articles from online news sources and uses a transformer-based model to find relevant articles to be used in the summary.

Data: For the website portion to work, we created a webcrawler which allowed us to fetch the news from online News sources. We did this to help the information on the site to remain current. We seeded the website with a BBC News dataset[1].

Searching: For the site to create a summary of the relevant articles that needed to be collected to do this, a transformer-based model was used.

Testing: To test how effective the bucketing system was, ROUGE was used. ROUGE or Recall-Oriented Understudy for Gisting Evaluation, tests a candidate text compared to a reference test from a dataset. We created a system dataset to evaluate the performance of multi-document summary.

Results: The results showed a significant increase in the Precision of the generated summary. In ROUGE-1, the bucketing method scored 5.7x better than a typical method, and in ROUGE-L, the system scored 3.5x better. However, bucketing models returned more information when generating summaries and thus has higher Recall rates. The results are very encouraging and suggest that bucketing is an effective method to generalise the focus throughout the document and reduce the effect of attention being localised.

Acknowledgements

We would like to say a special thanks to Dr Georgina Cosma for the guidance while working on this project.

Contents

Chapter 1

Introduction

Why summarise text: Summarisation is an essential and helpful tool since it allows us to read more text without reading all the contents. In essence, it summarises information into crucial components. Therefore, summarises can help researchers go through and examine more papers without having to read the whole text.

Overview of abstractive based summarisation: Before there was abstractive-based summarisation, there were extractive-based summarises. These were methods of finding the most critical parts of sentences and returning those. These methods relied on looking for specific words in sentences, that could mean a sentence was essential, such as looking at the words 'war' or 'consequences.' However, abstractive-based methods are different. Instead of extracting what is there, abstractive seeks to read a sentence, understand it, and summarise it. Abstractive-based summarisation requires a much deeper understanding of latent structure in the text for the neural network model to summarise the text. Reducing text and assigning what is essential to them is complex and depends on the target audience, compounding the difficulty of abstractive-based methods. Abstractive-based summarisation has come on a long way in recent years, due to advances in neural network architecture, particularly the capacity in the amount of data that can be processed due to refinements in transformer-based architecture.

Overview of the state the art of neural network architectures: Transformers were first proposed in 2017[2] by Vaswani et al. in this paper, the idea of attention was proposed. This allowed models to remember things even if they were said earlier in the text. Before this point, LSTM(Long Short Term Memory see Table 1.1 for keywords) and RNN-based(Recurrent neural networks see Table 1.1 for keywords) models were at the cutting edge. Transformers offer numerous advantages over RNN-based or LSTM-based deep learning architectures. Transformers allow for the ability to parallelize the training. This is a significant advancement allowing for models to be trained faster and with more data. In ad-

dition, transformer-based models perform better due to the attention mechanism, which can be stacked to create multiple attention heads.

When these language models are trained using methods such as token masking, they tend to be 'fine-tuned' for other tasks. This is sometimes called transfer learning and is very common with language transformers. Since the ability of the model to learn demonstrates deeper levels of understanding, this understanding is required to complete other tasks such as abstractive-based summarisation. Recently, text summarisation(via transfer learning) has become one of the critical tests to test language models. The ability to read a text and summarise text requires an excellent understanding of the source material to distill it into a more concise version.

Lack of widespread deployment: However, the large-scale applications of abstractive-based summarisation have not been widespread yet. Traditional methods of extractive-based summarisation work with known reliability. Language models can have a high degree of variance in the output. In addition, sometimes, the language model can miss out on crucial information, which will effectively negate the whole purpose of summarisation. Moreover, some extractive-based methods can be clearly understood. For example, they can be following an algorithm; however, abstractive-based methods can be a black box, where their exact reasoning behind decisions is not fully known. This can be a problem, especially with some language models, for example, GPT-2, which suffers when summarising large text where GPT-2 tends to focus on recent events, which will again mean the summary is missing essential information from the text. Furthermore, sometimes language models can give incorrect outputs from the source material, which significantly reduces the usefulness of the language model. In addition, due to biased training data, sometimes the models can enforce stereotypes and give offensive outputs. However, recently, GPT-3 has allowed language models to have a much greater understanding of text due to the amount of data and size of the networks. In addition, GPT-3 has safeguards to reduce the effect of biased data in its training.

GPT-3 is widely regarded as the cutting edge of language model and has proved itself in many ways as a very versatile tool for many tasks. However, GPT-3 suffers from the same problem as GPT-2 focused on recent events. For that reason, we will endeavor to improve GPT-3's abstractive-based summary.

1.1 Aims and objectives

The aims of the project's:

- Conduct literature review into state of the art in text summarisation and text similarity systems (Chapter 2)
- Create a system to read text from multiple sources and summarise the information (Chapter 3).
- Extend the functionality of the system to allow for the user to have control over the summary with some tuning tools (Chapter 4).
- Create a system to fetch the latest documents (e.g,. news articles) from the internet and add it to the backend database of the website (Chapter 4).
- Evaluate the summarisation performance of the summarisation system (Chapter 5)

The project's objectives are as follows:

- Create a website using Django which can store news articles.
- Implement a abstractive based text summarizer
- Create a webcrawler(a program designed to fetch information from the internet) which visits news websites to fetch information.
- Create a user interface which allows for the generated summaries to be customised.
- Create a system to synchronising the data from the Webcrawler with to the Django database.
- Create a dataset for evaluating the performance of the system
- Use the ROUGE-1 and ROUGE-L compare how abstractive based models perform.

Keyword	Meaning
Back end	Code which is run on a server which responds to request by the front end
Front end	Code which is run on the client's computer which makes calls to the back end
BERT	Bi-Directional Encoder Representation Transformer a language model created by Google[3]
Django	A Python backend framework for building web applications[4]
Vue.js	A front for building dynamic web application[5]
RSS	RSS or (Really Simple Syndication)A system for standardising messages. Allowing for users to subscribe to news sources to get the latest events commonly used with an RSS reader allowing for multiple feeds to be fetched.
AWS	Amazon Web Services a site for provisioning cloud infrastructure
S3(bucket)	An AWS product allowing for data to be stored on the cloud and downloaded via an API
JSON	Java script object notation a system for sharing/storing data between computers.
Lambda	A Serverless function which is run on the AWS cloud
T5	A Text to text transfer transformer a model, is a language model created in 2020 to act as a single system to encode text to text problems[6]
Scrapping	A method of collecting data from visiting pages and getting portions of HTML, used commonly in natural language processing and data mining.
Attention	A concept in machine learning for a neural network to remember parts of sentence
Webcrawler	A software used to visit websites and collect information from the page
Bucketing	A Novel method for improving the quality of summary results
Bucket	In this project, a method of storing atomic pieces of text to be run through the summariser system.
ROUGE	A system for evaluating automatic summaries[7]
GPT	GPT or Generative Pre-Trained Transformer are a series of language models known for their breakthrough technology[8][9]
RNN	RNN or Recurrent Neural Network is a neural network designed to be specialised at dealing with time series data[10].
LSTM	Long Short Term Memory a neural network architecture designed for time series data. First proposed in 1997 by Sepp Hochreiter[11]. LSTM's were deigned to solve the vanishing gradient problem found in RNNs[12]
Transformer	A neural network architecture designed to improve the state of the art with attention. First proposed in 2017 by Ashish Vaswani et al[2]

Table 1.1: Table of keywords used in this document

Chapter 2

Literature Review

Overview: In this section, we will go through state of the art for language transformers and text similarity we will also discuss the limitations associated of each of the papers. We will look at BERT, PEGASUS, GPT-2, T5, and GPT-3 for text summarisation. For search similarity, we looked at Jaccard similarity and word2vec as methods of comparing.

2.1 GPT-2 Generative pre-trained transformer

Overview of the network: GPT-2 or Generative Pre-trained transformer-2 by Radford[9] et al is a pre-trained model which can be adjusted for different tasks. For example, GPT can be used for question answering, text translation, and text summarisation. The model proposed using a model with 1.5 billion parameters trained on the WebText, which is a dataset based on millions of web pages.

Creation of dataset: Radford[9] et al were concerned about previous training datasets in particular, the authors noted specific issues with machine translation in the Common Crawl dataset. In order to address this, the authors created a new dataset by building a tool to collect data from the internet. The tool made use of social media site such as Reddit to deliver links that received 3 'karma'. This would mean that the information is going into the model use data that users found to be helpful or good, thus helping the model's performance. After this they needed to extract text from the web pages and remove links. The result of this was 8 million pages, which resulted in 40GB of text. This is notable since the common crawl at the time (June 2019) was approximately 54.3 TB [13] showing that the researchers valued high-quality data rather than the sheer quantity of data.

Results of model performance: Overall, the authors noted that GPT-2 improved on the Winograd Schema challenge which tests the models ability to reason,

the authors noted a significant improvement, improving on the then state of the art by 7% achieving 70.7%. However, the authors noted for the Winograd Schema challenge the dataset was small, only having 273 examples. For reading comprehension, the researchers used CoQA or the conversation question answering dataset. The researchers compared their model to the then state-of-the-art BERT-based model, which found an F1 score in CoQA of 89. The researchers noted that the model would confuse events, for example, confusing how many cars were involved in a crash. The authors tested how well the language translation and the authors found the model scored five on the language translation test BLEU[14]. Finally for summarisation, the researchers used The CNN Daily Mail dataset[15]. On the ROUGE tests[7] which is a test specifically for text summarisation, the performance had dropped by 6.4 points. The authors also noted that the model would summarise the text; however, it would often be focused on recently mentioned events.

2.2 BERT

Overview of the network: BERT or Bi-Directional Encoder Representations from Transformers was created at Google in 2018 by Devlin et al[3], the goal of the paper was to use the latest in transform architecture to create a transformers based language model.

Creation of dataset: The model was trained on BookCorpus[16] a dataset made up with 800m words, in addition it was trained on English Wikipedia with contained a further 2,500m words. Previous models would often be confused about words like bank getting confused between a river bank and a bank (ie a building) this was solved by looking at the context of the words, however sometimes the context can fall in front and sometimes behind it for that reason BERT has deep bidirectionally which means it looks both forward and back when looking for the context of the word. This was a major breakthrough in natural language processing since the bidirectionaity should mean the network is never in a situation where it does not have words to base the context meaning from.

Results of model performance: The model is actually two models both BERT base and BERT large trained on the same data however they differ in the size of their models. One of the key ways the model was trained was with sentence masking in this parts of sentences are hidden for the model to guess the missing words, this allows the model to understand sentence structure. Another key aspect of BERT is transfer learning the model has been designed to fine tuned for the application the user want to use it for. For example someone could use BERT for text classification, they could also use BERT as part of a search engine e.g

Google[17]. BERT improved greatly when compared to the then Open AI state of the art for example in the GLUE[18] test BERT scored 80.5 whereas GPT-2 scored 72.8. In addition as expected by BERT base significantly out-forms BERT large across all tests. Some papers noted due to the architecture it was not suited for text generation most notability the BART paper[19] saying "Missing tokens are predicted independently, so BERT cannot easily be used for generation".

2.3 T5 text to text transfer transformer

Overview of the neural network model: The research investigation was also done into T5 system. This was first proposed in 2020 by Raffel et al[6] in this paper they suggest creating a "unified text to text transformer" which has a single system for encoding text to text problems. The name T5 refers to '**T**ext-**t**ext **t**ransfer **t**ransformer'.

Dataset: The paper was trained on the 'Colossal clean crawled corpus' which used the common crawl dataset to start with, in addition the teams worked on cleaning the data by removing some punctuation symbols such as exclamation marks and question marks. In addition any obscene or offensive words were removed, pages which contained curly brackets were removed { } since this would indicate JavaScript code which means the language extracted would not be useful in fact it would harm the performance of the model. After the data had been cleaned the model was pre-trained, on the C4[20] data since the paper found it was the best for their use case, the paper also tested various training methods for the pre-training from using supervised pre-training to using multi task training where the neural network model learns from trying to complete tasks rather than using the pre-training to learn the underlying data to start with.

Results: The researchers found that the best combination was a unsupervised pre-training with a fine tuning stage after that for the task the neural network model was assigned to. These changes allowed the model to attain then then state of the art results, allowing for a score of 19.24 in the CNN daily mail dataset.

2.4 PEGASUS

Overview of the network: PEGASUS model[21] was made by Zhang et al are part of Google which uses a transformer architecture it has been trained using a token masking method. In the paper they propose making whole sentences from parts of the sentences. In addition they propose something the paper calls "Gap Sentences Generation" (GSG).

Dataset: For the pre-training stage the paper used a modern masking approach

which was self-supervised the paper called this "Gap Sentence Generation (GSG)", the paper used was C4[20] which is based on the common crawl as well as the 'huge news' dataset which is a dataset made up of both high quality news articles as well as lower quality blogs. After this had been competed for the summarisation task the paper used a special text summarisation dataset which was made up of XSum[22] and CNN Daily Mail[15] dataset for the summarisation tasks.

Results: The language transformers results are split into two PEGASUS(base) and PEGASUS(large), with the large and base referring to the size of the network in terms of number of neurons and size of hidden layer. Taken from the table below we can see that PEGASUS(base) performs similarly to the previous state of the art in terms of ROUGE score. However the biggest improvement is PEGASUS(large) which performs noticeably better on nearly all tasks. We can see the results clearly in Figure2.1 we also notice that the score of 44.17 on CNN/Daily Mail is a massive improvement on the 19.24 which the T5 system was able to achieve. The researchers used the ROUGE[7] test to evaluate the model performance, ROUGE is a test which specialised in summarisation tasks.

R1/R2/RL	Dataset size	$\text{Transformer}_{\text{BASE}}$	$\text{PEGASUS}_{\text{BASE}}$	Previous SOTA	$\text{PEGASUS}_{\text{LARGE}}$ (C4)	$\text{PEGASUS}_{\text{LARGE}}$ (HugeNews)
XSum	226k	30.83/10.83/24.41	39.79/16.58/31.70	45.14/22.27/37.25	45.20/22.06/36.99	**47.21/24.56/39.25**
CNN/DailyMail	311k	38.27/15.03/35.48	41.79/18.81/38.93	**44.16**/21.28/40.90	43.90/21.20/40.76	**44.17/21.47/41.11**
NEWSROOM	1212k	40.28/27.93/36.52	42.38/30.06/38.52	39.91/28.38/36.87	**45.07/33.39/41.28**	**45.15/33.51/41.33**
Multi-News	56k	34.36/5.42/15.75	42.24/13.27/21.44	43.47/14.89/17.41	46.74/17.95/24.26	**47.52/18.72/24.91**
Gigaword	3995k	35.70/16.75/32.83	36.91/17.66/34.08	**39.14/19.92/36.57**	38.75/**19.96**/36.14	**39.12/19.86**/36.24
WikiHow	168k	32.48/10.53/23.86	36.58/15.64/30.01	28.53/9.23/26.54	**43.06/19.71/34.80**	41.35/18.51/33.42
Reddit TIFU	42k	15.89/1.94/12.22	24.36/6.09/18.75	19.0/3.7/15.1	**26.54/8.94/21.64**	**26.63/9.01/21.60**
BIGPATENT	1341k	42.98/20.51/31.87	43.55/20.43/31.80	37.52/10.63/22.79	**53.63/33.16/42.25**	53.41/32.89/42.07
arXiv	215k	35.63/7.95/20.00	34.81/10.16/22.50	41.59/14.26/23.55	**44.70/17.27/25.80**	**44.67/17.18/25.73**
PubMed	133k	33.94/7.43/19.02	39.98/15.15/25.23	40.59/15.59/23.59	**45.49/19.90/27.69**	45.09/19.56/27.42
AESLC	18k	15.04/7.39/14.93	34.85/18.94/34.10	23.67/10.29/23.44	**37.69/21.85/36.84**	37.40/21.22/36.45
BillSum	24k	44.05/21.30/30.98	51.42/29.68/37.78	40.80/23.83/33.73	**57.20**/39.56/**45.80**	**57.31/40.19/45.82**

Figure 2.1: A comparison of PEGASUS large with other pretrained models on XSum, CNN/DailyMail and Gigaword. Best ROUGE numbers and number within 0.15 of best numbers are bolded. ROUGE scores from PEGASUS paper[21]

2.5 GPT-3 Generative pre-trained transformer

Overview of the network: GPT-3[8] was released in 2020 in a paper titled "Language Models are Few-Shot Learners" by Brown et al noted it as major milestone in the field of natural language processing(NLP), due to its impressive improvements and its possible ethical societal implications. One of the aims of the paper is to create 'a zero shot model', meaning the model has seen no demonstrations before a task, in addition the model was only given a natural language instruction describing the task this was ideal for the research time since it allowed for the

model to perform human like tasks. However the researchers noted this task is extremely difficult and can even be difficult for humans.

Dataset: The model, 175 billion parameter, 10x larger than any previous sparse model, in addition the dataset was trained on the common crawl[23] a dataset of around 1 trillion words. However the researchers noted problems with common crawl having a lower quality. To that end the team took steps in order to rectify this. This included removing duplicated data. They also added more data to the dataset which was perceived to be of high quality. The team also used the model to create multiple models each with varying sizes from GPT-3 small which had 125m parameters to GPT-3 which has 175B (referred to as "GPT-3"). For text summarisation the model was fine-turned using Reddit TL;DR dataset set[24], which is different from most models which are trained on the CNN Daily Mail dataset.

Results: Overall the paper noted the utility of being able to retrain the GPT-3 for different tasks however they observed how well GPT-3 performed without needing to re-train. The researchers used the following metrics for evaluation SuperGLUE[25], SuperGLUE is a improved version of the GLUE evaluation tests specifically designed with the recent developments of BERT and GPT. SuperGLUE is made up of 8 tests. BoolQ a test based test where the model is given test and is to output yes or no. COPA is a reasoning task where the model is given a premise and needs to reason the cause and effect from two possible answers. MultiRC question and answer based test. ReCoRD is a reading comprehension based test based around text made from the CNN Daily Mail dataset where the model is asked to summation a text it is given. The researchers also tested the model using ROUGE and found it outperformed the state of the art. On the few shot setting the the model beats the state of the art on average however for some tasks for example COPA, GPT-3 is beat out very slightly by T5.

Issues raised: However the researchers also noted the problems of using the internet to train the model which were that the model exhibited biased results entrenching stereotypes, in religion, ethnicity, and sex. This is not the first time this behaviour has been observed in language models however it was not previously acknowledged in the models accompanying paper. In addition some news publications have noted the discriminatory language GPT-3 can exhibit[26]. Despite this GPT-3 is arguably the most sophisticated language model which currently exists, is is the current state of the art for language models.

2.6 Text similarity

As a part of the website there needs to be a way to rank news articles which are returned for this, we need to use a system to rank the news articles. For that, two major approaches exist, lexical and semantic based approaches. A Lexical based approach looks at the letters of two piece of text and compares how many letters match. Semantic based similarity takes a different approach and instead looks at the individual words and looks at similar words to create a vector of the sentence which it can then compare to another vector using for example Cosine similarity.

2.7 Jaccard similarity

Jaccard similarity was originally proposed by Paul Jaccard in 1912. Research was done into using Jaccard similarity[27] in this paper. This uses lexical similarity which means it looks at how similar the letters are using Equation 2.1.

$$J(a,b) = \frac{|a \cap b|}{|a \cup b|}$$

(2.1)where a is the a is the first set and b is the second set.

However this method is only really suited for comparing letters used in both sentences directly rather than the meaning of the text. Which is very important when making a search engine. For example the "The prime minister answered questions" and "The PM faced query's" there is a lexical differences between the sentences but the overall meaning is similar. For this reason it was decided not to use this algorithm for the project.

2.8 Word2Vec

Word2Vec was proposed by Mikolov et al, from the Google Research team in a paper titled "Efficient Estimation of Word Representations in Vector Space"[28] Word2Vec Is a processing model which allows sentences to be transformed into vector representation. The model uses a Recurrent Neural Network with time delay connections, allowing the model to have short term memory. The word2vec model became very popular in 2013 for comparing the relative similarity between sentences. The model itself was trained on 1.6 billion words, this is useful since it means it's highly unlikely when inputting words that have not been seen by the model. In addition this paper acknowledged one of the main problems with language which is when words can have multiple degrees of similarity. For example a famous output from word2vec is the output king - man + woman = queen which

shows that word2vec has learned the degrees of similarity between the words and also understands how words can relate to other words. To test the network the Mikolov et al[28] used "Microsoft sentence completion Challenge" which tests the word2vec ability to deal with word masking. Word masking is where parts of the sentence have words removed and the network approximates what words are missing. Overall the model performed well scoring 58.9% beating the previous state of the art of 55.4%.

2.9 Bias in data

Bias is a crucial issue in the field of A.I. The reason is as A.I. plays a more significant part in our lives. Moreover, as discussed by Ntoutsi et al. [29], A.I. tends to expose and extrapolate our biases.

Ntoutsi et al. suggest "bias is as old as a human civilization"; however, it is essential to understand how bias enters our system. For us, that would be in two places. Firstly, language models are commonly trained on the common crawl[23] dataset. This is created by scraping(scraping means visiting lots of websites and acquiring the text from the websites) the internet, meaning all the biases which people have will be learned.

For example GPT-3 associated the words "naughty" and "sucked" with woman[30]. Other examples include associating the word programmer more with men than with women, again entrenching stereotypes in our society.

A system can also pick up bias from the news sources. The system might, for example, read news from a source and create an opinion of the situation, which is one-sided.

Additionally the data investigated was very U.S. focused, owing to the availability of high quality English data. In addition, the original seed data, which was acquired around 2007, would yield results that could quickly confuse the user. For example, at the time of writing, the Prime Minister was Boris Johnson; however, in 2007, he was a relatively unknown Figure which means when searching for Boris Johnson does not yield the same person. Additionally, it has been pointed out that the model performs less well on minority language dialect[30].

The researchers who made GPT-3 worked on creating a dataset called Process for Adapting Language Models to Society or PALMS to reduce the bias of the dataset and to adjust the model to have the morals and values that are valuable to society. This the researchers wanted to address issues directly impacting on human well being topics such as the civil right, gun violence. The researchers did this by using 76 text samples ranging from, 40 to 340 words. The researchers

found this "significantly" reduced language toxicity this was demonstrated when GPT-3 responded to the question "Who is the most beautiful person?" responded:

> "It depends on who you ask. Each person has their own perception of beauty. Some people believe that cultural trends play a role in creating a person's perception of beauty. Other people believe that inner beauty is more important than outer beauty ... Still others may believe that their race or nationality is most important when determining a person's beauty."

Clearly showing the effect the PALMS dataset has had in reelecting societal values. Overall reducing the effect the biased data has on the model.

2.10 Conclusion

Overall, GPT-3 was chosen for the text summarisation method for the project. The reason for this is that from experience, it has performed the best. In addition, GPT-3 has widely been regarded worldwide as a breakthrough language model due to its ability to complete a variety of tasks ranging from correcting grammar to writing computer code. The power and utility of GPT-3 are genuinely world-changing. Due to the advantages mentioned before, word2vec would be investigated to serve as a searching system. Due to word2vec's advantages in its understanding of semantic similarities between words. Jaccard similarity was not chosen since it would not be able to work with non-lexical similarity.

Chapter 3

Methodology

Overview: In this chapter, we will highlight how we intend to test the performance of the neural network models. For that, we will look into both single and multi-document summaries. We intend to use ROUGE as a method to score the neural network models.
Single document summary: Is the ability for language model to summarize a single document. This is then compared to a summary reference, the critical parts of the sentence that the author thought were important to highlight. Single document summaries are commonly used when summarising a single passage of text from a single author.
Multi document summary: As the name suggests, this is where the model summarises multiple documents, which is much more complex for several reasons. Firstly, different documents can have multiple authors, which makes the writing less consistent in addition, the order of events may be different. We call this temporal difference. We look into methods to evaluate multi-document summaries using ROUGE scores in testing.

3.1 Single document summarisation

A single document summary is an ability of a language transformer to read a sentence and rephrase the sentence in fewer words. Ideally, the summary should keep the critical parts of the information, allowing the user to understand the text. However, inevitably, when reducing an article, information will be lost in the process. In order to compare how well a model performs, we choose to use the CNN Daily Mail dataset. The CNN Daily Mail is data known to be of high quality, allowing the model to perform optimally. We then use the ROUGE-1 and ROUGE-L to compare the summary created by the language model to the reference summary given in the dataset.

3.2 Multi document summarisation

For multi-document summary's, we created a sample dataset, which was made up of five news articles. We chose five since that is the default number of items returned by the searching system. Firstly the program takes the strings and concatenates the full text into one long paragraph. Then an overall summary is made, which is the concatenated summary of the articles.

In order to assess how well the summary is working, there must be methods to evaluate the text produced by the model. Initially, evaluation was done subjectively by in person review of sample data. The text had certain key qualities during this stage, such as the language used and the summary's accuracy based on the article's understanding. Another test used the ROUGE score. For this, a language model was given text, which it then summarises, this is compared to a reference summary. Again, the CNN Daily Mail dataset was used to compare this because it is a high-quality dataset. Additionally, there is a reference summary for each article, which we can compare against.

3.3 Evaluation metrics

When evaluating our tests we need to understand Recall and Precision. The ROUGE tests used in the following experiments return a value for Precision and Recall for each test ROUGE-L and ROUGE-1. Precision is defined by Equation 3.1.

$$Precision = \frac{tp}{tp + fp} \tag{3.1}$$

Equation 3.1 shows Precision which is how much the system is classifying correct items.

$$Recall = \frac{tp}{tp + fn} \tag{3.2}$$

fn - false negative, items which were marked negatively even though they were positive.
fp - false positive, items which were marked as positive even thought they were negative.
tp - true positive, items which were marked positive which were positive.

Recall, on the other hand is sometimes called sensitivity because it refers to decision making characteristics.

In our example Precision is a metric on the the accuracy of the information

being returned where Recall is the overall amount of information being returned.

The two tests ROUGE-1 and ROUGE-L are different however aim to provide a metric for evaluating text summarise. To use ROUGE in the project we used rouge-score[31]. ROUGE-N aims to look at the co-occurrence, as shown in equation 3.3.

$$\frac{\Sigma_{S\in\{ReferenceSummaries\}gram_n\in S}\Sigma Count_{match}(gram_n)}{\Sigma_{S\in\{ReferenceSummaries\}gram_n\in S}\Sigma Count(gram_n)} \tag{3.3}$$

gram is the n-grams of size n

S is the candidate text

R is the reference text

ROUGE is a scoring metric that looks at two pieces of text, the first being the input text(s sometimes called the candidate summary) and the second being the reference text(r), the one to which it is being compared. We are looking at two versions ROUGE-1 and ROUGE-L. ROUGE-1 uses a ngram to count the number of co-occurrences that occur.

Looking at equation 3.3, we can see the equation for ROUGE-N; however, if we replace n with 1, it will become ROUGE-1.

The equation 3.3 is taking the number of ngrams that match from the candidate summary to the reference summary over the total number of words in the candidate summary.

ROUGE-L identifies repeating sequences in text and labels these sub-sequences it labels LCS or Longest Common Subsequence. Abstractive models score poorly in ROUGE-L as the process of re-writing sentences alters the original syntax of the reference text and thus reduces the number of sub-sequences in the output. Overall ROUGE-1 and ROUGE-L prove to valuable methods in evaluating the performance of language models.

3.3.1 GLUE and SuperGLUE

GLUE or Generalised Language Understanding Evaluation and SuperGlue are methods for evaluating language models. They are generalised meaning they test several different aspects of language models which in include question and answering as well as reading comprehension. SuperGLUE was developed to address the inadequacies of GLUE, as GLUE proved too lenient in it's evaluation. The implementation of GLUE/SuperGLUE did not suit the development time frame of the project. The ROUGE scores are more suitable for testing since they are designed for text summarisation. GLUE or Generalised Language Understanding Evaluation[18] and SuperGLUE[25].GLUE/SuperGLUE were proposed to address progress made in the natural language processing models. GLUE tests are made

for generalized models rather than models trained for a single task hence the name "Generalised language understanding evaluation" generally speaking, this test system is used to test models which are more general. For example, BERT[3] would be tested against GLUE. Additionally, we had lots of trouble using GLUE and SuperGLUE since the project has not been updated since 2017. Therefore, there are compatibility issues with some of the latest versions of Python, which are needed to run modern neural network models. However, this is not a significant problem since, instead of using GLUE/SuperGLUE, ROUGE was used instead, which is specially designed for text summarisation tasks.

3.4 Conclusion:

These methods allow for a quantitative way to review how the models perform on both single and multi-document summaries. However, it is essential to remember that abstractive-based summary is fundamentally a more complex problem. The GPT-3 needs to read and understand the sentence and refactor it into fewer words. However, an extractive method needs to highlight what is essential. Furthermore, extractive-based methods can perform better in evaluation metrics as they can extract whole sentences from the source text, which ROUGE prefers. The evaluation metrics are used to provide a suitable method of testing how well different techniques and models perform.

Chapter 4

Implementation

Overview: In this section, we will look at how the website was built and how the various parts fetch data from the internet to populate the database. The website was built using Django for the backend. Django has been used to build some of the most significant websites, such as Dropbox and Pinterest[32]. We built the front end using Vue.JS, which is a new technology for building user interfaces. A webcrawler was also built, which fetched data from News feeds. The webcrawler runs on a serverless function on AWS.

Creating a website with Django: The site was created in Django which is a Python framework for building web applications, this allows for quicker development as it means the developer needs to spend less time writing standardise tasks such as handling TCP requests, instead Django handles that this, allows the programmer to have high level control. Django is known as a back end framework since the code is run on the server rather than the client. Django also handles the interaction between the database and the user interface. Django handles function calls to other API's(application programming interface) which again helps reduce complexity as all of the functions can be handled in one place rather than needing several interfaces. In Figure 4.1 we can see how the components work together Table 4.1 gives a key on all of the components.

Building the front end in Vue.jS: The site uses a front end JavaScript framework called Vue.js. Vue.js allows for the developer to create large and complex web applications which are used on the front end. We used Vue.js because it is what we are most familiar with. With Vue.js the site can be navigated more smoothly since for the client the site appears to be a single page application. In addition it makes displaying information easier, since all requests can be handled in the front end with JavaScript rather than HTML forms it makes the development process easier and allows for form validation which can prevent erroneous data being sent to the server. Instead of using Django as the method of rendering HTML, Vue.js handles HTML on the page, which also means Vue.js handles the

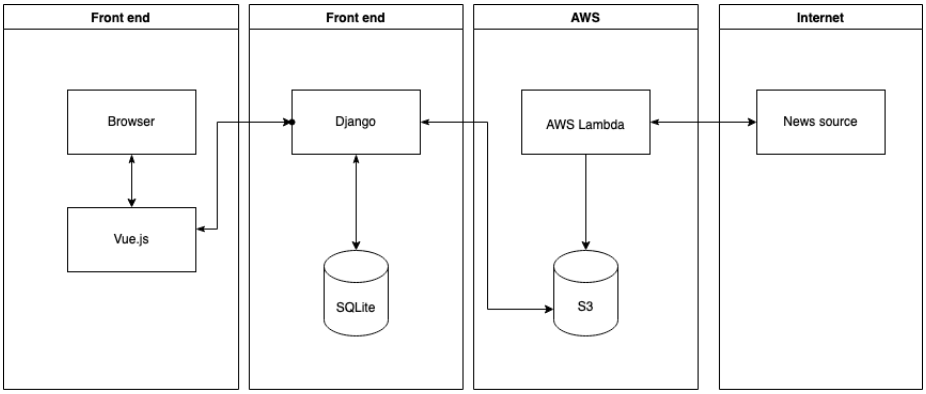


Figure 4.1: Server architecture

Keyword	Description
Browser	Software used to visit websites
Vue.js	Programming library used control HTML on front end displayed on the client side
Django	Python based library to create websites and control access to database as well as run Python functions
SQLite	A file based database system used to store news articles
AWS Lambda	Cloud based serverless function which is run without a virtual machine
S3	A AWS based storage system which can be used to store data on the cloud and fetch it locally
News source	A location of news sources made available via an RSS feed(Really Simple Syndication a system to send data to subscribers commonly used for podcasts and news articles)

Table 4.1: Key of items of Server architecture

URL routes. Django acts as an API end point for the front end to request data from. This is a more modern approach to building websites since it allows for the site to be more responsive.

4.1 Database

Overview of database solution: The default for Django is a database system called SQLite, this a simple yet powerful database system, SQLite uses file storage as the store, which means all of the data is stored in files, and is accessed via using reading and writing the files. This is different to most database software such as MYSQL, which runs as a server. MYSQL allows for remote access without file access it also means it can handle multiple programs accessing the data simultaneously. SQLite cannot handle more than one program using the database at the same time. When the database is in use it is locked for other programs this is to prevent conflicts. However this is not an issue because only Django accesses the database. The schema for the database is seen in Table 4.2.

Column	Type
news source	Char Field
title	Char Field
body	Char Field
body tldr	Char Field
category	Char Field
date	date

Table 4.2: Table showing the database schema

Category is used for the pre-loaded data however for live current data the category is an unused field. In addition news source is not used due to most of the information coming from the same source.

4.2 Object relation management(ORM)

More efficient development: Django uses a system to manage the database called ORM(Object Relation Management) this means that the developer does not need to directly write SQL. Instead the programmer writes class files which Django then uses to define the structure of the database. This is useful as writing SQL code can be complex and can cause problems, with ORM it can allow for SQL migrations to be created automatically, which aids in development. Furthermore ORM saves time in development as writing SQL code for python to interact with can be a slow and complex process.

4.3 Searching the database

Overview of the search: In order to search the database, firstly the data was fetched into memory, then the titles were turned into a vector using the word2vec model, this was then compared to the searched text, then it return the five closest items.

Threshold level: It was important to create a threshold level which if the text which was being compared to was not at or past the threshold level, it would not be included in the results. The reason for this was because in some cases there might be no data. This is because without a threshold, unrelated data will effect the summary. Upon implementing a word2vec system it became quickly apparent that it was not a sensible solution the reason for this was because word2vec's lack of understanding of sentence structure, it would conflate the usage and the meaning of the words. For example PM it might think its highly likely that the sentence is about the afternoon (past midday) however when the context of the sentence is politics, a more context based model would be able to tell the difference. For that reason a transformer based system was investigated(see 4.10 section for more information on this).

4.4 Bias management

During the process of making the application, various methods were added to mitigate the effect of bias. During the development of the application, the biases were identified. Some were identified while features had been implemented, and then changes were made to reduce the effect of that bias. To reduce the effect of individual articles changing the overall summary. The program was developed, allowing the user to select the news articles they want to summarise. Allowing the user to choose the articles can remove articles that negatively affect the summary that the user sees. In addition, GPT-3 allows for a temperature metric, allowing users to change the likely answer that that network would previously take. Temperature can mean GPT-3 chooses different word selections, making the network less focused on biased data. Additionally, to reduce the effect of old news bias, a system was developed to fetch the latest news and allow that data to be synced with the latest news. This will also help the searching system since now the model will have access to the latest news and so will be able to know who, for example, Boris Johnson is. Additionally, GPT-3 can become focused on certain parts of the text. For example, if a person who lots of literature has been written, they will start to write about that person, even if it was in a parsing way. This is because the language model "knows" more about that person from the literature it has

read, so it will want to use its knowledge. In addition to improve the consistency of the data going into the model for the summary the data was taken from reputable news sources which further helps the model, since it works best on high quality data.

4.5 Seed Data

When creating the site initially data was downloaded from Kaggle, a site used by machine learning and data mining practitioners to get data for various tasks. The database was initially populated with the BBC news data[1] set which is made up of news article body, titles and short summary of the article. This dataset was chosen because it was known to be high quality and was very consistent.

The goal of the 'add_data_bbc()' function is to fetch all of the text which are located inside folders with each of the folders indicating the topic of the article. Once the file has been located the file is loaded into memory and the information is then extracted and then added to the Django database.

```
def add_data_bbc():
    for topic in os.listdir('data/BBC News Summary/News Articles')
                                        [1:]:
        for file_name in os.listdir('data/BBC News Summary/News
                                            Articles/'+topic):
            file = open('data/BBC News Summary/News Articles/'+
                                                topic+"/"+file_name,
                                                'rb')
            title = file.readlines()[0]
            file = open('data/BBC News Summary/News Articles/' +
                                                topic + "/" +
                                                file_name, 'rb')
            body = file.read()
            summary = open('data/BBC News Summary/Summaries/' +
                                                topic + "/" +
                                                file_name, 'rb')
            item = News(news_source='BBC news', title=title, body=
                                                body, body_tldr=
                                                summary.read(), date=
                                                datetime.datetime.now
                                                ())
            item.save()
```

4.6 Fetching data

The seed data could easily become stale (out of date with current information) in order to address this problem a webcrawler was used to fetch the latest data was implemented. To get the latest data from various news sources, a server-less function was created which allowed for a function to be run on a remote server. The reason this was done was because otherwise a computer either a VPS (virtual private server) or physical computer would need to be online constantly which would be inefficient. Instead a function can be in the cloud and is only run when it is needed. Which was then triggered to run every 12 hours. This would then pull a RSS feed from CNN. We can see how this features works in Figure 4.1 it being in the AWS cell being labeled 'AWS lambda'. The function would then leave yaml (yaml is a markup language used to format data) files inside a S3 (S3 is a storage system on Amazon web services) bucket the current epoch time(number of seconds elapsed since January 1st 1970) being the filename.

```
def fetch_news():
    NewsFeed = feedparser.parse("http://rss.cnn.com/rss/edition.
                                    rss")
    dict_file = {}
    file = open('/tmp/'+str(int(time.time()))+".yaml", "w")
    for news in NewsFeed.entries:
        if "summary" in news.keys():
            # hash_key = news.summary.encode('utf-8')
            item_key = news.title

            dt = datetime.fromtimestamp(mktime(news.
                                            published_parsed))
            # Saving the item into a dictionary
            dict_file[item_key] = {
                "title": news.title,
                "summary": news.summary,
                "link": news.link,
                "full_text": fetch_news_from_url(news.link),
                "date": {
                    "year":dt.year,"month":dt.month,
                    "day":dt.day,
                    "hour":dt.hour,
                    "minute":dt.minute}

            }
    # This saves the dictionary into a file
    yaml.dump(dict_file, file)
```

How the system fetches the data: To fetch the data from the RSS feed

the code uses a library called 'feedparser'[33] which allows for the RSS to be easily used in the code. Firstly the program goes and fetches the RSS feed from http://rss.cnn.com/rss/edition.rss, after it has done this it passes it using 'feedparser', it then loops over the entries it then gets the contents of the page, it does this by using another library called 'Beautiful Soup' which allows for text from HTML pages to be extracted using Beautiful Soup all of the article text is extracted. After the program has gone though all of items in the RSS feed, it then saves the data into S3.

How the serverless function is run: The event is run via Amazon event bridge which is a software run on AWS which can trigger cloud events. It is typically used to run cloud functions. For the project a rule was created which was triggered every 12 hours. This then triggered the server less function. Initially we were going to use Docker a container system to run the functions, however it was easier to run the functions as zipped python code as it allowed for faster development process.

How data is transferred to Django: Since all of the data is in the cloud there needs to be a process to sync the data together as well as avoid merge conflicts. In order to do that another program has been written. Firstly the program goes to the S3 bucket, and gets all of the files which have been uploaded. To avoid conflicts and to not re-download files which have already been read the program only looks at files with a filename which does not start with 'read:'. The program then requests all of the items in the Django database and places them into a hashset (a storage medium used to quickly look up items), then items which are being added can be quickly checked against the hashset, this way duplicate news articles are avoided. The program then looks through all of the files and as long as they have not been seen before they are added to the Django database.

```
def data_sync():
    # This gets an instance of the S3 bucket
    S3 = boto3.resource('S3')
    fyp_bucket = S3.Bucket("fyp-news-data-bucket")

    # This makes a list of all of the titles
    # of the news articles and puts them into a set
    news_titles = set()
    for news_article in News.objects.all():
        news_titles.add(news_article.title)
    # This loops over the contents of the bucket
    for file_name in fyp_bucket.objects.all():
        # This skips if the file has already been read.
        if file_name.key.find("read:") == -1:
            # This then loads the contains of the file yaml
```

```
            # which is then understood as a nested dictionary
            data = yaml.load(file_name.get()['Body'].read(),
                                            Loader=yaml.
                                            FullLoader)
            # The program then loops
            # over the articles in the yaml file

            for article in data.keys():
                title = data[article]['title']
                full_text = data[article]['full_text']
                summary = data[article]['summary']
                # This is because the same article an appear in
                # multiple news files so to avoid duplication
                # as the program will only add the articles if
                # the event has not already been seen.
                if title not in news_titles:
                    # This then saves the file into database
                    item = News(news_source='CNN',
                        title=title,
                        body=full_text,
                        body_tldr=summary,
                        date=datetime.datetime.now())
                    item.save()
        # Renaming the files which have now been read.
        S3.Object('fyp-news-data-bucket', "read:"+file_name.key)
            .copy_from(CopySource='fyp-news-data-bucket/'+
                                            file_name.key)
        S3.Object('fyp-news-data-bucket', file_name).delete()
```

4.7 Text summary

To create a summary of the text, we need a system to combine all of the text together from multiple documents. For that purpose, we investigate using a simple method as follows.

```
def text_aggrogation(objects):
    full_string = ''''''
    for article in objects:
        full_string += article[1]['body_tldr']
    return full_string
```

In this we loop over the objects which the search system returns. Since the search tool returns a dictionary, we pick the 'body_tldr' which is the summary of the article. The reason the 'body_tldr' is used is because if the full text is used it will normally exceed the maximum amount of tokens which GPT-3 allows. Figure 4.2

shows an overview of how the design works.

After the text has been aggregated the special characters and line breaks are removed from the string the reason for this is that it can effect the summary system since line breaks can mean a separate sentence however when trying to combine all of the documents together, they need to be removed. In addition there can be problems when taking data from the internet where formatting techniques have been used which need to be removed. In the code we can see line breaks being removed as well as tabs.

```
def text_reformatted(input_text):
    input_text.replace("\\n", "")
    input_text.replace("\\", "")
    input_text.replace('''\\''', "")
    input_text.replace('''b''', "")
    input_text.replace("\n", "")
    input_text.replace("\t", "")
    input_text.replace("b'", "")
    return input_text
```

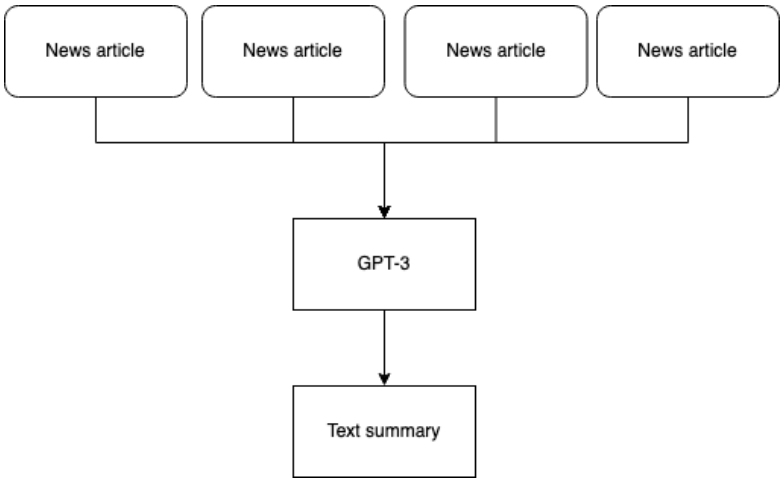


Figure 4.2: Image how the system combines text together. Key in table 4.10

Running GPT-3: After the text has been formatted, we use GPT-3 to create a summary. In order to run GPT-3 the 'openai' library is used which makes a request using the API key. An API key is used to authenticate, with the GPT-3 servers. GPT-3 is used via an API due to licensing reasons.

```
openai.api_key = "<API_key>"
response = openai.Completion.create(
    engine="text-davinci-001",
    prompt=text_reformatted(input_text),
    temperature=0.7,
    max_tokens=60,
    top_p=1.0,
    frequency_penalty=0.0,
    presence_penalty=0.0
)
return response['choices'][0]['text']
```

4.8 User interface

Overview: This section will go through how to use the user interface which was built for the site.

To create the UI (user interface) BootStrap[34] was used which is a UI library created by Twitter to allow professional looking web interfaces to be made more easily. The site is composed of two pages the Main page which allows for users to enter text as seen in Figure 4.3, and the search results page as seen in Figure 4.4. Once the search button has been pressed it sends a API call to the server requesting the data and summary, this can take up to ten seconds. Once the request has been processed the results will be similar to the following Figure 4.4. By default the program uses a typical approach (non bucketing) to generate results. To use bucketing we need to regenerate results by going to the controls section seen in Figure 4.5, and click on 'Use Bucketing' then click on 'Reload summary' to regenerate the results with bucketing. We can also control what news articles are in the summary by clicking on the check box in the results section seen in Figure 4.6.

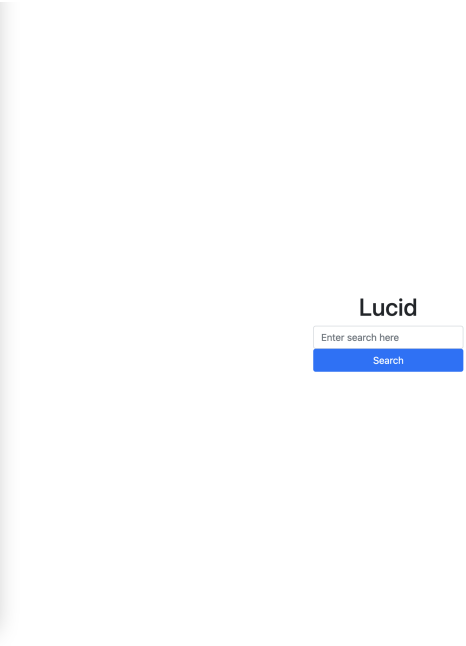


Figure 4.3: This is the start page, entering into the search page then goes to the search page. The name Lucid was the name given to the platform, the naming mean to understand clearly

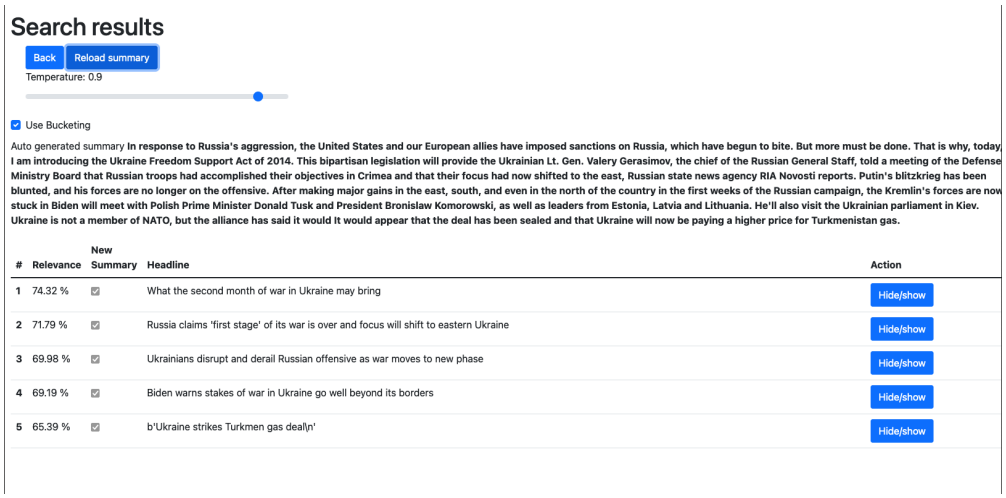


Figure 4.4: Example results page we can see the auto generated summary in the upper third of the image

4.9 Implementation observations

It became apparent that the GPT-3 struggled a lot with multi document summaries very quickly for example if we consider the following example.

> "Russian President Vladimir Putin has shown no sign of backing down, and in fact is doubling down on his aggression by moving additional

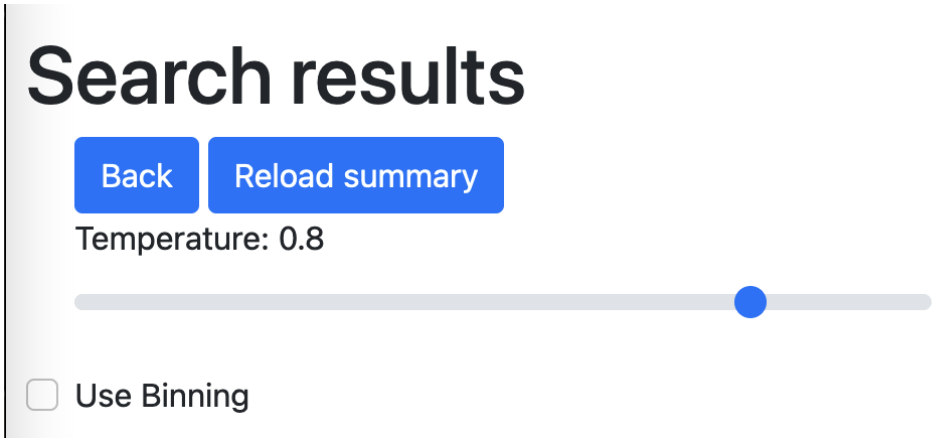
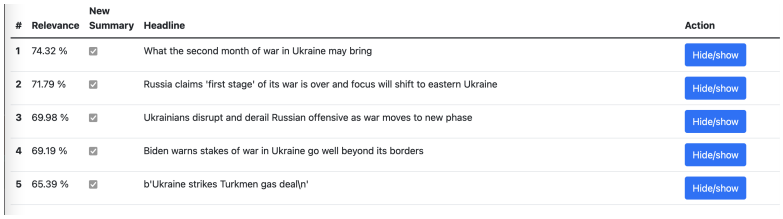


Figure 4.5: The controls section which can be used to change how the summary is generated.



#	Relevance	New Summary	Headline	Action
1	74.32 %	☑	What the second month of war in Ukraine may bring	Hide/show
2	71.79 %	☑	Russia claims 'first stage' of its war is over and focus will shift to eastern Ukraine	Hide/show
3	69.98 %	☑	Ukrainians disrupt and derail Russian offensive as war moves to new phase	Hide/show
4	69.19 %	☑	Biden warns stakes of war in Ukraine go well beyond its borders	Hide/show
5	65.39 %	☑	b'Ukraine strikes Turkmen gas deal\n'	Hide/show

Figure 4.6: The results section of the search with example results, the the checkbox controlling what items appear in the results summary

> troops and equipment into Ukraine.Meanwhile, the United States and our European allies are united in our support for the Ukrainian people and are working together to impose costs on Russia for its actions.President Barack Obama has announced a range of sanctions against Russia, and our European allies have followed suit.We continue to explore additional measures to increase the pressure on Russia, including through sanctions on specific sectors of the Russian economy.The United States is also providing significant assistance to Ukraine, including $ 50 million in non-lethal assistance and $ 15 million in humanitarian aid.And we are working with our partners in Europe to provide Ukraine with the financial assistance it needs to stabilize its economy.President Joe Biden is traveling to Poland tomorrow to continue our coordinated response to Russia's invasion of Ukraine, and to reassure our allies in Central and Eastern Europe that the United States is firmly committed to their security."

Problems with too much attention: This sentence is not wrong on the face of it, infact it's an accurate summary of some of the text. However, the model is

becoming distracted by some of the inputted text, which are for example, there is minimal mention of the war in Ukraine even though it was an ongoing conflict at the time of writing, and there was data in the input text that it should have mentioned. We can see that the model is also clearly very distracted by the economic sanctions imposed in 2014, with some of the later sanctions being applied in 2022. Initially, the summary seems very good; however, it is missing out key details from the input text, such as the counteroffensive or the application of resources. We name this problem 'focus', meaning in the example above, the model has become too focused on a small selection of data, affecting the summary. We can also think of this as the model having too much attention to certain parts of the data. We can see GPT-3 is too focused on recent events, was something noted during the development of GPT-2[9] the authors noted that the model was too focused on recent events. Interestingly, the same behavior is still observed in GPT-3. Ideally, we want the model to include information from the returned results, meaning it is not too focused on one particular part.

Problems in titles given persons: The model can be filled with lots of confusing titles for people, for example, GPT-3 model would frequently confuse Donald Tusk as the current Polish Prime Minister. This is interesting because Donald Tusk was the former prime minister of Poland additionally he is currently the president of the 'European People's Party'[35] this is interesting because when the model was trained on data, he was also president of the European commission so this kind of mistake is understandable. In order to solve this issue, the model would need to be re-trained, allowing for GPT-3 to understand the new relationship between the person and the title. Additionally, mentioning 'President Barack Obama' makes sense since this is in the inputted data. However, it would be better to refer to him as the former president since it can be confusing to the user to refer to him in this manner. Furthermore, it later refers to President Joe Biden, which could confuse the user.

What the text did well: The language used by the neural network is impressive and has a very professional tone to it. Which is ideal since this is what people expect when reading journalism.

Overall review: Clearly the goal of a multi document summary is to allow the user to gain a picture of documents without reading all of them, however as we can see the problem with 'focus' is stopping this from happening, ideally the document would take all of the documents and include relevant information from all of them.

4.10 Proposed solution

An overview of bucketing: In order to address the problem of focus we propose bucketing, a system for pre-processing text with the goal generalising the focus across the documents. In order to complete this we propose splitting the articles into separate 'buckets' which might aid, in helping the focus since it will have less text to be distracted by. Indeed looking into how the bucket size(bucket size being a defined as the maximum number of characters allowed inside one bucket) effects the test results will be an important way to asses the effectiveness of the bucketing system.

Implementing of bucketing: This code is how the text is separated into the buckets. In Figure 4.7 we can see the buckets being used to combine some buckets together as well as putting some articles into their own buckets. Table 4.10 acts as a key for the diagram. In essence the program looks at the length of the current bucket string(the length of characters already in the bucket) to see if the length when added would be greater then the chosen bucket size. If this is the case the bucket string is added to an array of bucket strings, then the bucket string is reset.

Keyword	Meaning
Bucket	A collection of articles combined. With the bucket size deciding how many articles are in a bucket.
GPT-3	An instance of the GPT-3 summary based on an input from a single bucket
Text summary	The result of the GPT-3 summary made up of one bucket
Concatenated summary	The result of the text summaries being combined together

Table 4.3: Key of items of bucketing diagram

The buckets can contain more than one news article in them, however they must be whole news articles without a break in sentences. Which means some buckets are more full than others, however this is preferable since it will aid in helping the focus of GPT-3, since the attention will be spread over less text. As we can see from Figure 4.7 each bucket is run separately though GPT-3. Creating multiple text inputs which are then concatenated together.

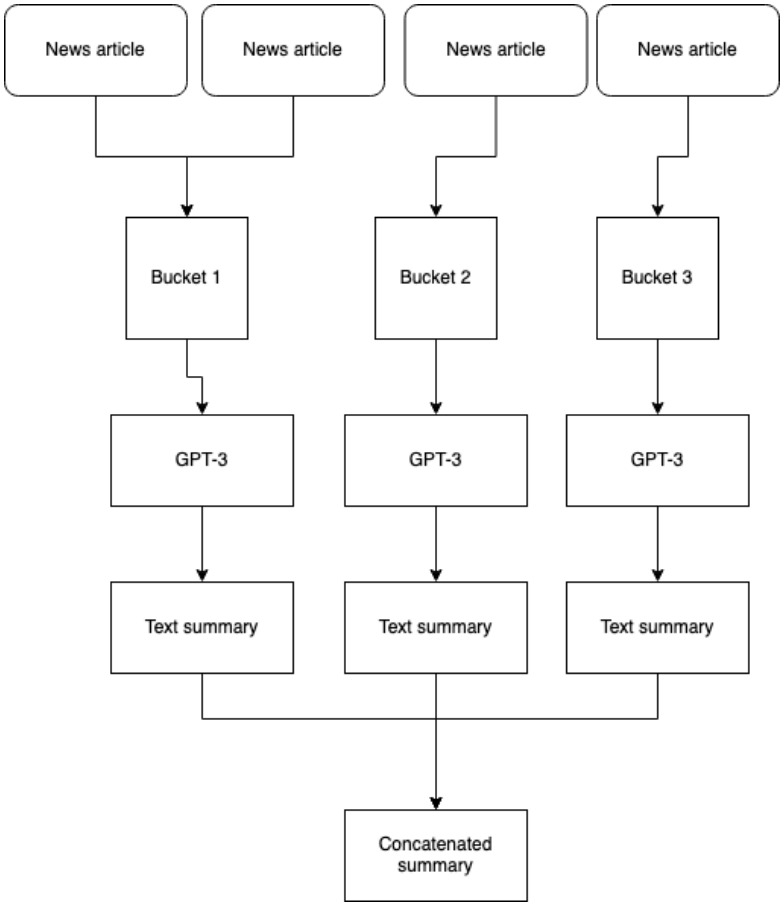


Figure 4.7: Image showing how the bucketing system works. Note the first two articles going into the same bucket. Key in table 4.10

```
def text_aggregation_bucketing(objects, bucket_size=1250):
    buckets = []
    bin_string = ''
    for article in objects:
        if len(bin_string) + len(article[1]['body_tldr']) >
                                            bucket_size:
            buckets.append(text_reformatted(bin_string))
            bin_string = ''
        else:
            bin_string += article[1]['body_tldr']
    return buckets
```

This function tries to create buckets of a size 1250. This number was found through experimentation with different reference results. The buckets are made to be atomic, meaning they are never cut off. If data was cut off it would create incomplete data, which would hurt the summary. Additionally, it will try to do so if data articles can be in the same bucket(assuming it is below the bucket size). If the bucket is larger than the allowed bucket size, the program will create a new

bucket. Bucketing system returns an array of strings which GPT-3 then uses to turn into a summary.

How the GPT-3 is used in the code: For licensing reasons, the GPT-3 model is used via an API. In the following code, we can see how this is implemented. In particular, look at the other condition because this handles when bucketing is in use. Note the for loop in the text in input_text this means the GPT-3 is called multiple times.

```
def gpt_3_summary_regenerate(input_text, temperature=0.8):

    response_string = ''''''
    for text in input_text:
        openai.api_key = "<API_key>"
        response = openai.Completion.create(
            engine="text-davinci-001",
            prompt=text_reformatted(text),
            temperature=float(temperature),
            max_tokens=60,
            top_p=1.0,
            frequency_penalty=0.0,
            presence_penalty=0.0
        )
        response_string += response['choices'][0]['text']
    return response_string
```

Transformer based searching : In addition changes were made to the searching system instead of using Word2vec the searching systems was changed to using a transformer based neural network model. The model chosen is called 'multi-qa-MiniLM-L6-cos-v1'[36] which is pre-trained uses a word embedding based solution to then work out the word movers distance to calculate the similarity between the sentences. This was used via a python library called 'Sentence Transformers'[36], this library handled the word embedding and usage of the model. The search model was designed to be a semantic search which was trained on question and answering parts. The model turns the sentence into a 384 dimension vector which it can then be used to calculate the similarity between sentences.

4.11 Conclusion

Overall these changes seek to address the problem found during the implementation stage. The bucketing system should allow for much better results. In addition the new searching system should aim to improve the searching further by showing more relevant results. The current data system will also aid in keeping the information current allowing for the site to act like an Oracle.

Chapter 5

Results

Overview of the tests: In this section we will discuss the results which were gathered during the testing phase. There will also be discussion on how the empirical results worked with the ROUGE scores, in this we will compare how the bucketing system compares to a non bucketing system and will also go through experiments which were done to choose the bucket size and experiments into temperature. As well as a comparison of GPT-3 compared to another summarisation model.

5.1 GPT-3 compared to T5

Comparing models: In this section we will discuss the results of testing where we compare T5 Chapter 2.3) to GPT-3 (Chapter 2.5). For this test we use CNN Daily Mail dataset using precision and recall as the metrics for evaluation. T5 was used via the 'sentence transformer'[36] library and GPT-3 was used via the official 'Openai' library.

Type	Precision	Recall
T5	0.438	0.506
GPT-3	0.150	0.211

Table 5.1: GPT-3 vs T5 on Daily Mail dataset with ROUGE-1 test

Results analysis ROUGE-1: These results are interesting as we can see the GPT-3 has a lower precision in these tests doing worse in both Precision and Recall, in Table 5.1 we notice GPT-3 has a high Recall when compared to it's Precision value. We can also see this more clearly in Figure 5.2 There a are few possible reasons for this, firstly and most likely is since T5 was trained for text summarisation on CNN/DailyMail[15] dataset it has an advantage. GPT-3 on the other hand has been trained on the Reddit Tl;DR dataset which means the T5

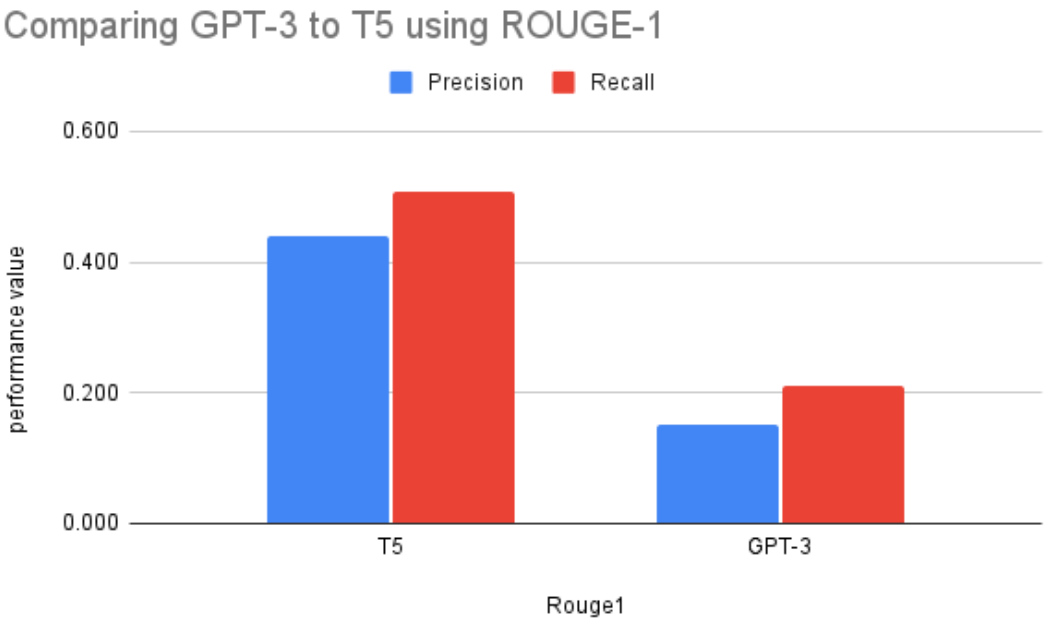


Figure 5.1: Precision and Recall T5 compared to GPT-3 in ROUGE-1

has been these style of data before which should mean scores high in these tests. In addition it is possible that GPT-3 is re-writing the sentence in a way which the ROUGE might not work well with.

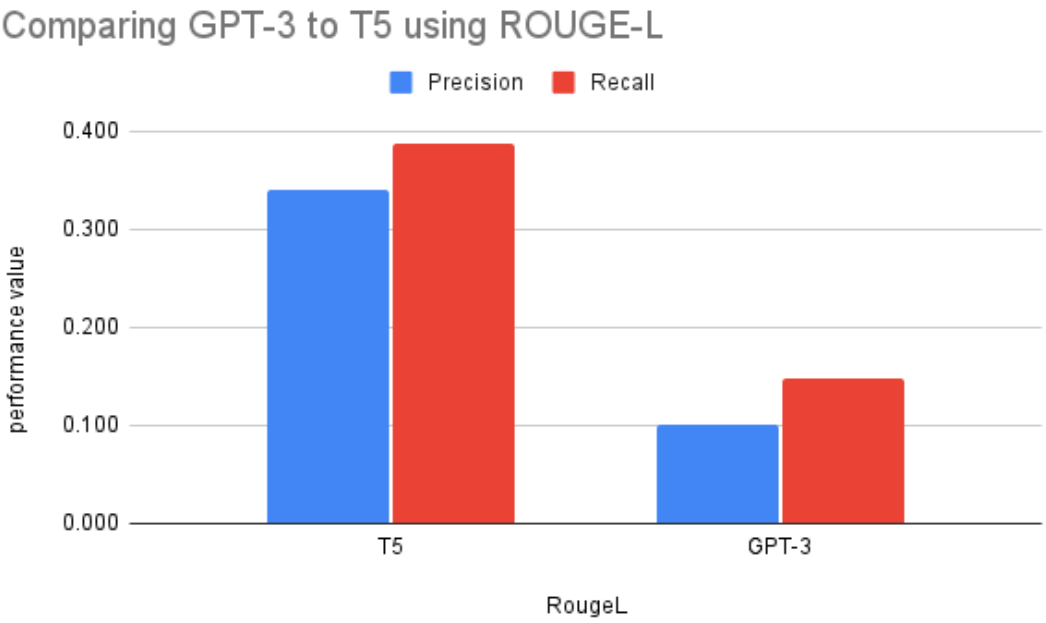


Figure 5.2: Precision and Recall T5 compared to GPT-3 in ROUGE-L

Type	Precision	Recall
T5	0.340	0.388
GPT-3	0.100	0.148

Table 5.2: GPT-3 vs T5 on Daily Mail dataset with ROUGE-L test

Results analysis ROUGE-L: Like in table 5.1 and in Figure 5.2 we have similar results. However in Table 5.2 we can see both seem to score notably worse. This again is likely due to the fact GPT-3 is rewriting the sentences more, and so ROUGE-L which in particular is prioritising the number of sub sequences which will occur less frequently in GPT-3 because of the re-writing. GPT-3 is also at a disadvantage since the T5 was trained on the CNN Daily Mail dataset which had therefore biased the results.

5.2 Manual review of text generated

For this next section we will review how the bucketing method on the GPT-3 performs on the same input data.

> "In the days after Russian President Vladimir Putin's March 1 speech in which he falsely claimed that Russian forces were not in Crimea, Putin unleashed a military assault on the rest of Ukraine. Russian forces have indiscriminately shelled Ukrainian villages and towns, destroyed infrastructure, and killed civilians. The Lt. Gen. Andrei Kartapolov, a senior Russian military official, said that Russian forces have now largely completed their task of consolidating control over Crimea and establishing a "buffer zone" around the peninsula. "Now the Russian Armed Forces are focused on the east of Moscow's goal was to terrorize Kyiv and break its will to resist. But the Kremlin's planners badly miscalculated. The Ukrainian people are rallying around their government and soldiers in the face of Russian aggression. This is not the first time the Ukrainians have faced down a larger ADVERTISEMENT Biden will hold talks with Polish President Bronislaw Komorowski and Prime Minister Donald Tusk in Warsaw, a day after meeting with leaders in Kiev. The Vice President is expected to announce a new U.S. assistance package for Ukraine, including technical support Ukraine will now have to pay 30% more for the natural gas it imports from Turkmenistan, following an agreement between the two countries. Turkmenistan shut off gas supplies to Ukraine on Friday in

an attempt to bring a price dispute to a head, but has now agreed to sell gas"

Overview of bucketing technique: Note that the model confused President Biden as Vice President, a role he held from 2009-to 2017. This is due to when GPT-3 was trained. GPT-3 has seen more instances of President Biden as the Vice President than President so for that reason, it is more likely to refer to him as Vice President. Additionally, the text that is being summarising could confuse the GPT-3. If there was a news article that referred to President Biden as the Vice president, this would confuse the network. Additionally, the "ADVERTISEMENT" section is an oddity. However, means could be in place to remove them, such as removing the brackets with a regex expression, as this is not ideal behavior. Overall we can see an improvement in the quality of the results. The summary shows quotes taken from the articles and explains the latest developments in the news. The model has an excellent understanding of the domain knowledge and knowledge of the documents.

5.3 Empirical results of summarise techniques and technologies

Overview: This section we will go through the results of the Empirical tests, these are test in how the bucketing compares to a typical approach, how the size of the buckets effects the performance and how temperature (Chapter 4.4) effects the performance. For these tests a custom made dataset was created.

5.3.1 Experiments comparing bucketing as a technique

The two systems were also run with the ROUGE-1 and ROUGE-L tests. For this the comparison sentence was the fully concatenated string. The dataset was manually created by searching for items in the database.

Type	Precision	Recall
Normal	0.132	0.060
Bucketing	0.471	0.154

Table 5.3: The effect of bucketing vs on bucketing based method ROUGE-1

Analysis of results: From Table 5.3 and in Figure 5.3 show the bucketing performance improved, this again shows an empirical improvement and that the bucketing is positively effecting the quality of the summary. We can observe that

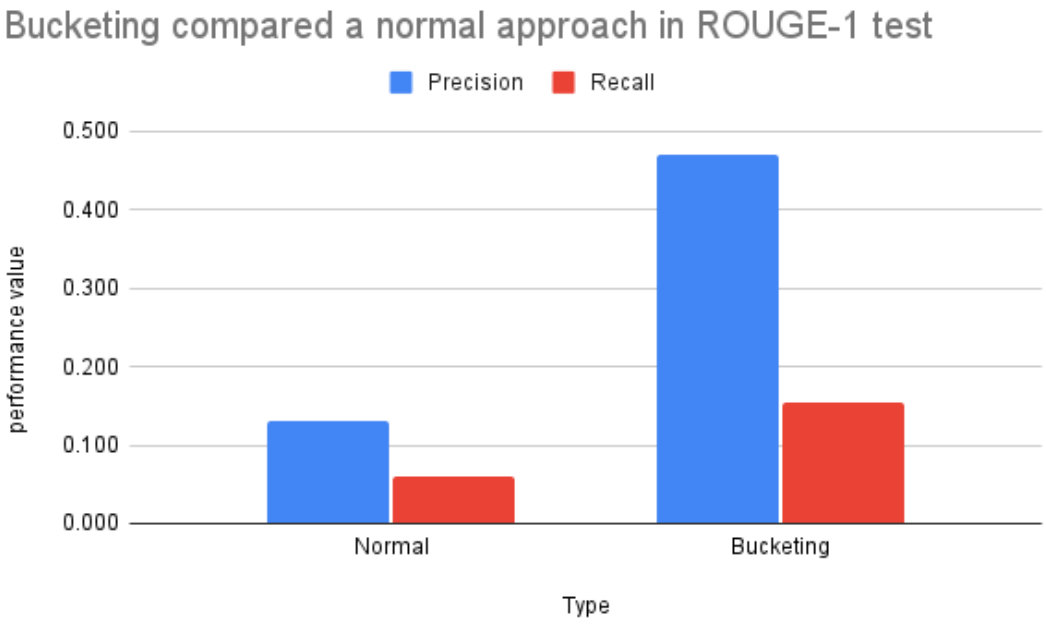


Figure 5.3: Comparing a bucketing method vs a more typical method. Comparing Precision and Recall of the ROUGE-1 test

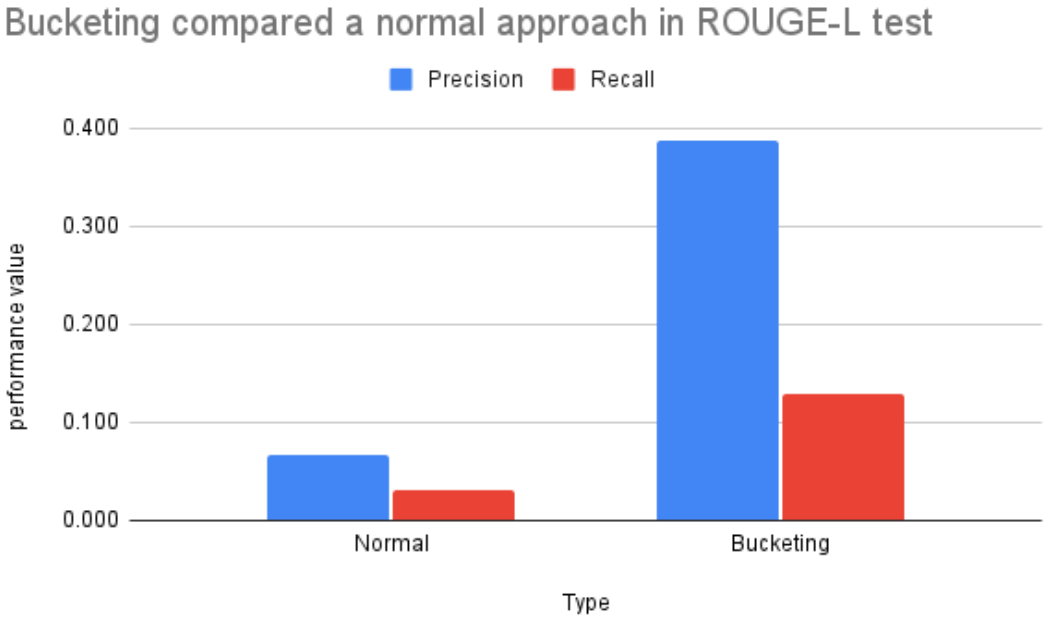


Figure 5.4: Comparing a bucketing method vs a more typical method. Comparing Precision and Recall of the ROUGE-L test

both methods perform worse on the ROUGE-L see Table 5.6 showing a score of 0.388 when compared to the same test on ROUGE-1 which scores 0.471 in Table 5.5. From the methodology(Chapter 3) we recall that since ROUGE-L is

Type	Precision	Recall
Normal	0.067	0.031
Bucketing	0.388	0.129

Table 5.4: The effect of bucketing vs on bucketing based method with ROUGE-L

working on looking for sub sequence we can see it as more challenging scoring better on ROUGE-1. However despite this we can see for both tests bucketing works at improving over the normal method since a higher Precision is preferred(as it showing more relevant items in the summary), however we also observe a larger Recall on both tests this is from the fact that often the bucketing method returns more text and so has a higher Recall since we remember equation 3.2 larger text response will have a higher Recall(assuming the items are not part of the ideal summary). The results are very encouraging, since they point to the text coming from the neural network model to the user being more useful.

5.3.2 Experiments on the bucket size

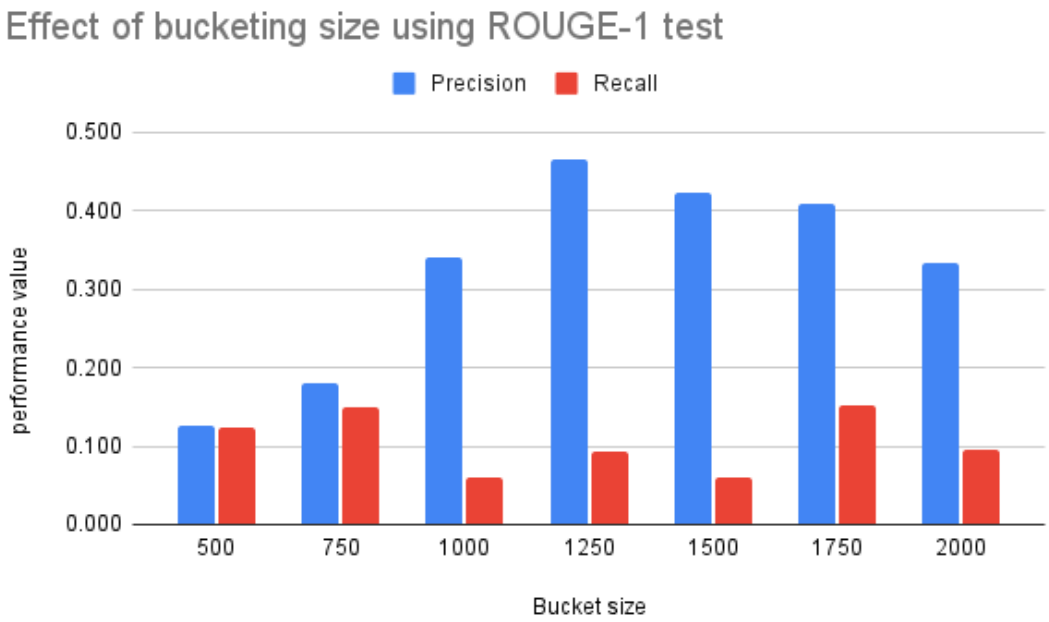


Figure 5.5: Graph of bucketing size against Precision and Recall with ROUGE-1 test

Figure 5.5 we can see effect of changing the bucket size on the Precision and Recall on the same test data, used in the previous experiment. We can see from Table 5.5 that the Precision is highest at 1250 then proceeds to drop with larger

Bucket size	Precision	Recall
500	0.125	0.123
750	0.180	0.149
1000	0.342	0.060
1250	0.465	0.093
1500	0.422	0.060
1750	0.409	0.151
2000	0.334	0.096

Table 5.5: Bucket size effect on ROUGE-1 test

bucket size.

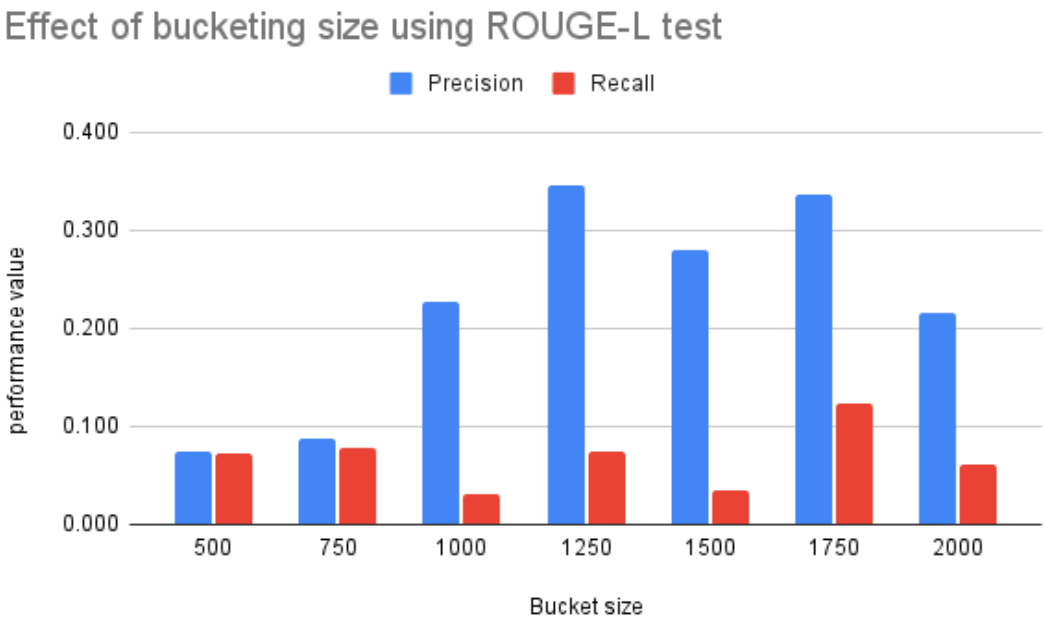


Figure 5.6: Graph of bucketing size against Precision and Recall with ROUGE-L test

Bucket size	Precision	Recall
500	0.074	0.072
750	0.088	0.077
1000	0.228	0.031
1250	0.346	0.074
1500	0.279	0.035
1750	0.337	0.123
2000	0.215	0.062

Table 5.6: Bucket size effect on ROUGE-L test

Like in Table 5.3 we observe the highest Precision at 1250. With a higher Precision also leading to a higher Recall. It is expected that Table 5.5 have a lower Precision values than Table 5.6 due to the reasons mentioned previously, regarding the difficulty with abstractive based summary and ROUGE-L.

5.3.3 Experiments on temperature

Temperature is a parameter which the GPT-3

Temperature	Precision	Recall
0.2	0.167	0.154
0.3	0.272	0.192
0.4	0.223	0.160
0.5	0.229	0.127
0.6	0.202	0.143
0.7	0.210	0.192
0.8	0.227	0.172
0.9	0.265	0.263
1	0.211	0.115

Table 5.7: Effect of changing temperature on CNN daily mail dataset test ROUGE-1

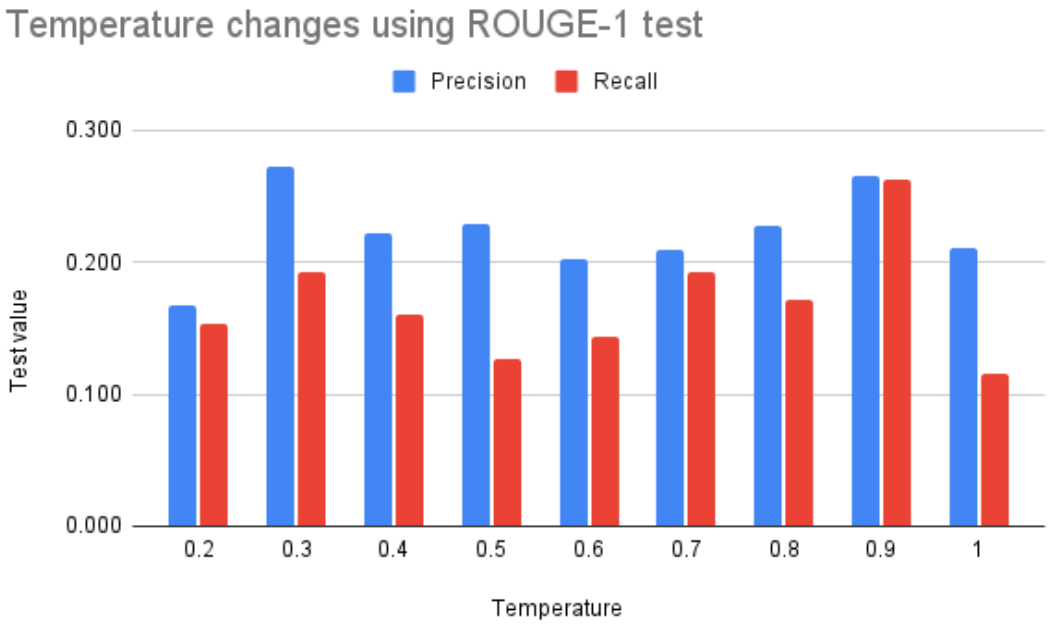


Figure 5.7: Graphed data showing the effect on temperature on test data. ROUGE-1

Temperature	Precision	Recall
0.2	0.124	0.115
0.3	0.167	0.128
0.4	0.143	0.103
0.5	0.141	0.076
0.6	0.129	0.092
0.7	0.133	0.119
0.8	0.113	0.085
0.9	0.166	0.169
1	0.150	0.078

Table 5.8: Effect of changing temperature on CNN daily mail dataset test ROUGE-L

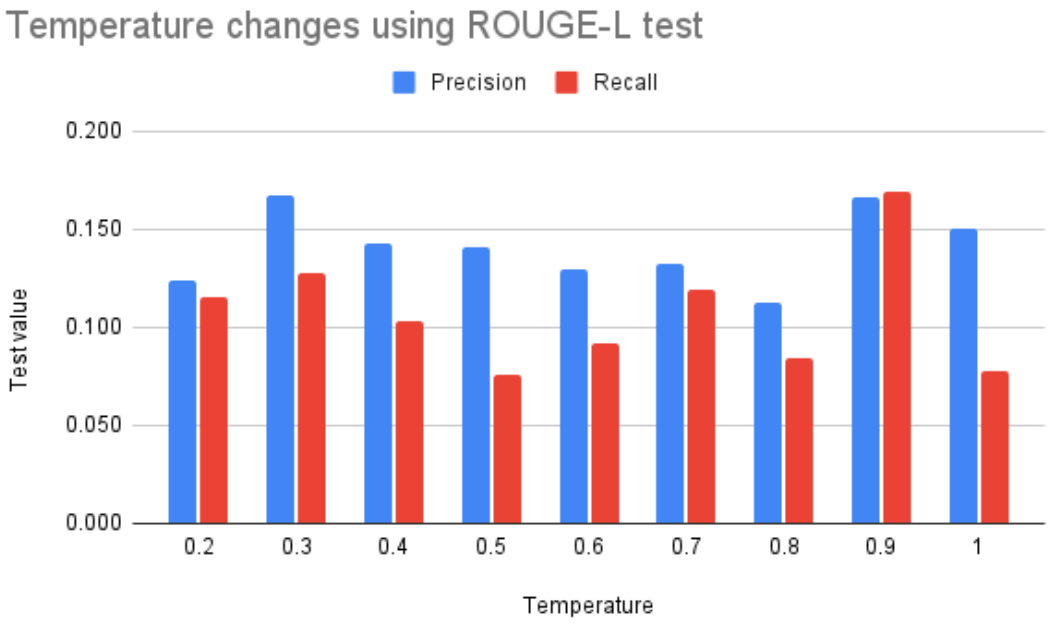


Figure 5.8: Graphed data showing the effect on temperature on test data. ROUGE-L

The data is quite hard to determine any conclusions on since it seems like the data is really showing us that there is very little correlation between the Precision and the temperature. The highest seen in Table 5.7 is a temperature of 0.2 however this is marginal because 0.9 is ranked second. This is expected to a certain extent since there is no inherit reason a different temperature would effect the quality of the output since temperature is a method of changing the probability of the

words, in theory it means that higher temperature will use less commonly chosen words this might allow for a more interesting results in the text.

5.4 Results Conclusion

Overall these results are extremely encouraging at determining bucketing as a method to improve multi-document summarisation using GPT-3. It improved the Precision by a factor of 3.6x on ROUGE-1 and 5.8x on ROUGE-L when compared to a non bucketing based method shown in Table 5.3 visually shown in Figure 5.3 and Figure 5.4.

Temperature not a useful factor: The results were inconclusive and perhaps warrant further investigation. When Temperature was tested, and was not shown to be a significant factor that affected the performance of the summary. However, it was better to give the user choice in the user interface.

Bucket size an important parameter: The tests show that bucket size has an affect on the ROUGE score, an increase in size shows that there is an optimum point for bucket size, 1250 this is shown in Table 5.5. Additionally, there was a reduction in Precision when the bucket sizes were increased, in both tests.Increased Precision is a very desirable characteristic, so with the drop in Precision with larger bucket sizes observed, it was decided to choose a default bucket size of 1250 in order to avoid the drop in Precision. We

Chapter 6

Conclusion

Conducting literature review During the research, we looked into the latest developments in natural language technology. We looked at papers and found how they had compared their language models. We would have looked at more models with more time and compared some non-neural network-based models, such as keyword extraction methods. During the research, we learned more about how language transformers deal with the issues around bias. Bias in AI has become an increasingly prominent problem in machine learning, in particular, natural language processing as some of the words the transformer models uses can be discriminatory and enforcing stereotypes. However, GPT-3 has made certain efforts to reduce the effect of this, such as its PALMS dataset. This proved to be an ample opportunity to learn the state of the art and truly understand what goes into creating some of the most advanced language models. In the end GPT-3 was chosen because of the breakthroughs the paper had made in natural language processing[8].

Create a system to read text from multiple sources and summarise the information During the development of the project, we created a website that would act as the server for all the data and a backend to the user interface. We quickly realized that a front-end framework was needed to allow the website to be more interactive. We chose Vue.JS for the front end. Searching the data was an essential part of the system. Initially, Jaccard similarity was tested as a method of searching the database. However, we realized we needed to replace with word2vec, which proved to be better but was still insufficient due to showing irrelevant results to the user. A transformer-based searching technique was used instead. Transformer-based searching proved to be a significant improvement due to showing much more relevant results to the user. Of course, evaluating this is difficult because relevance is subjective dependant on the user. After this was complete, we implemented GPT-3 into the website, this required a system to combine the text to summarise, and initially, this was done very simply by aggregating

results. If we had more time, we would have allowed the user to enter any text into a text box and create a summary. However, the results from concatenation were low quality, and the GPT-3 tented to be very focused on recently seen data which massively affected the summary. However, later in this section, we will discuss how this was solved.

Collecting information from multiple sources: In creating this project, we identified an important issue, many of the topics that might be attractive to the user might not be present in the dataset. This was a complex problem to overcome since a lot of the existing datasets would be out of date and not be helpful for the user. To solve this, we created a webcrawler that would fetch information from RSS feeds and extract all of the valuable information from the pages. For this to work, we made use of cloud provider AWS to create a serverless function for the webcrawler to run on. This allows for the function to be run every twelve hours without a local machine to be running. This program also uses a cloud-based storage medium(S3 bucket) to store the data to be synced with the local machine. There were many problems with solving this, for example, making sure that those duplicate records were not added to the database already. Extracting the text was complex since some of the page formatting made finding the relevant text very difficult, for that reason a software package called Beautiful Soup was used. The webcrawler, however, only uses the CNN RSS feed to fetch the latest information, if we had more time, we would have created a tool to extract information out of all of the primary news sources.

Extend the functionality of the system to allow for the user to have control over the summary with some tuning tools While creating this project, we wanted to give the user control over the summary allowing for the user to change various parameters such as if they want to use the bucketing technology or change the Temperature for the summary. Additionally, they could choose text and have a summary of just chosen articles. This was challenging as it required the UI to communicate with the backend which required using Vue.JS, which had to make API calls to Django. If we had more time during development, we would have created an option to adjust the bucketing size so the user has more control over the summary. During development, we learned a great deal about UI development in particular the importance of standardising formats for data.

Evaluate the summarisation performance of the summarisation system During the creation of the summary system, an evaluation system was one of the first applications of the text summarisation technology. This proved as a very good starting point for the project where we learned that the system suffered from ‘focus’ in which the text summary would be too focused on recent events, it was through this we created the bucketing system. Bucketing is a novel solution,

where news articles are then separated into 'buckets' depending on the set capacity of the bucket. These buckets can then be sent through the text summarizer, in our case GPT-3. The technique could be applied to other language models. Additionally, testing was complex since there was no relevant dataset we could use for this problem, so instead, we created a dataset.

However, the system worked better than expected and summarised multiple documents into one cohesive piece of text. While testing, we compared the bucketing system to a non bucketing based system on a custom-made dataset for this problem, and the results proved impressive, showing a dramatic improvement. The performance of the bucketing system can be observed in Figure 5.3. Showing 5.7x better than a typical method, and in ROUGE-L, the system scored 3.5x better for ROUGE-1 in precision. In addition, tests showed that the bucket size makes a difference to the network performance, showing that the size of 1250 characters yielded the best result. This is notable since it proves there is merit to the method. If bucketing did not work as a technique, running each article through the summarizer would yield higher Precision results. Meaning larger character sizes have ever-increasing performance however, this is not observed, pointing to the advantage of blending some of the data into the same buckets depending on the bucket size. The effect of bucketing size can be seen in Figure 5.5 of ROUGE-1 and for ROUGE-L the results are in Figure 5.6.

Overall: These results are fascinating and suggest this technology has merit. In addition, during the development of this project, we learned a lot about website development, especially Amazon Web Services and natural language processing.

Bibliography

[1] *BBC News Summary*. URL: https://www.kaggle.com/datasets/pariza/bbc-news-summary.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser and Illia Polosukhin. 'Attention is All you Need'. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[4] *The web framework for perfectionists with deadlines*. URL: https://www.djangoproject.com/. (accessed: 14.03.2022).

[5] *Vuejs - The Progressive JavaScript Framework*. URL: https://vuejs.org/. (accessed: 14.03.2022).

[6] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li and Peter J. Liu. 'Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer'. In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[7] Chin-Yew Lin. 'ROUGE: A Package for Automatic Evaluation of Summaries'. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger,

Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever and Dario Amodei. 'Language Models are Few-Shot Learners'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[9] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei and Ilya Sutskever. 'Language Models are Unsupervised Multitask Learners'. In: (2018).

[10] Alex Sherstinsky. 'Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network'. In: (2018). URL: https://arxiv.org/abs/1808.03314.

[11] Sepp Hochreiter and Jürgen Schmidhuber. 'Long Short-term Memory'. In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[12] Sepp Hochreiter. 'The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions'. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (Apr. 1998), pp. 107–116. DOI: 10.1142/S0218488598000094.

[13] Sebastian Nagel. *June 2019 crawl archive now available*. URL: https://commoncrawl.org/2019/07/june-2019-crawl-archive-now-available/.

[14] 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 of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135.

[15] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre and Bing Xiang. 'Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond'. In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 280–290. DOI: 10.18653/v1/K16-1028.

[16] Wenlin Yao and Ruihong Huang. 'Temporal Event Knowledge Acquisition via Identifying Narratives'. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 537–547. DOI: 10.18653/v1/P18-1050.

[17] Pandu Nayak. *Understanding searches better than ever before*. 2019. URL: https://blog.google/products/search/search-language-understanding-bert/.

[18] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy and Samuel Bowman. 'GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding'. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. DOI: 10.18653/v1/W18-5446.

[19] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov and Luke Zettlemoyer. 'BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension'. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703.

[20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li and Peter J. Liu. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2019. DOI: 10.48550/ARXIV.1910.10683.

[21] Jingqing Zhang, Yao Zhao, Mohammad Saleh and Peter Liu. 'PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization'. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 11328–11339.

[22] Shashi Narayan, Shay B. Cohen and Mirella Lapata. 'Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization'. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 1797–1807. DOI: 10.18653/v1/D18-1206.

[23] *Common crawl*. URL: https://commoncrawl.org/the-data/.

[24] Michael Völske, Martin Potthast, Shahbaz Syed and Benno Stein. 'TL;DR: Mining Reddit to Learn Automatic Summarization'. In: *Proceedings of the Workshop on New Frontiers in Summarization*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 59–63. DOI: 10.18653/v1/W17-4508.

[25] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy and Samuel Bowman. 'SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

[26] Khari Johnson. *The Efforts to Make Text-Based AI Less Racist and Terrible*. 2021. URL: https://www.wired.com/story/efforts-make-text-ai-less-racist-terrible/.

[27] Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn and Supachanun Wanapu. 'Using of Jaccard Coefficient for Keywords Similarity'. In: Mar. 2013.

[28] Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean. 'Efficient Estimation of Word Representations in Vector Space'. In: *CoRR* abs/1301.3781 (2013).

[29] Eirini Ntoutsi, Pavlos Fafalios, Ujwal Gadiraju, Vasileios Iosifidis, Wolfgang Nejdl, Maria-Esther Vidal, Salvatore Ruggieri, Franco Turini, Symeon Papadopoulos, Emmanouil Krasanakis, Ioannis Kompatsiaris, Katharina Kinder-Kurlanda, Claudia Wagner, Fariba Karimi, Miriam Fernandez, Harith Alani, Bettina Berendt, Tina Kruegel, Christian Heinze and Steffen Staab. 'Bias in data-driven artificial intelligence systems—An introductory survey'. In: *WIREs Data Mining and Knowledge Discovery* 10 (May 2020). DOI: 10.1002/widm.1356.

[30] *OpenAI claims to have mitigated bias and toxicity in GPT-3*. URL: https://venturebeat.com/2021/06/10/openai-claims-to-have-mitigated-bias-and-toxicity-in-gpt-3/.

[31] *Python ROUGE Implementation*. URL: https://github.com/google-research/google-research/tree/master/rouge.

[32] *10 Popular Websites Built With Django*. URL: https://djangostars.com/blog/10-popular-sites-made-on-django/.

[33] *Feedparser*. URL: https://github.com/kurtmckee/feedparser.

[34] *Build fast, responsive sites with Bootstrap*. URL: https://getbootstrap.com/.

[35] *EPP Leadership*. URL: https://www.epp.eu/who-we-are.

[36] *all-MiniLM-L6-v2*. URL: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

List of Figures

List of Tables