Blurble - Abstractive Novel Summarization

Theo Delettre
School of Computing
Dublin City University
Dublin, Ireland
theo.delettre2@mail.dcu.ie

Albertas Ovodas
School of Computing
Dublin City University
Dublin, Ireland
albertas.ovodas2@mail.dcu.ie

***Abstract*—We consider the application of document summarization models to full book texts, with the objective of automatically generating a book's blurb. This presents unique challenges not only in the great length disparity between a book and its summary but also in the literary style of the source text. We adapt and experiment with existing language models fine-tuned on various labelled training sets. The produced summaries are evaluated by comparing them to sample human written summaries through the standard ROUGE metrics. While some methods produce promising summaries, they run into the limits of the ROUGE evaluation metrics. Further developments in abstractive summarization of long-form text would greatly benefit from an improved evaluation method. We also explore the use of NER models in identifying the main characters in a book. While NER models are adept at identifying character names, they lack the ability to deduce character importance or the multiple names of a single character**

***Index Terms*—Natural Language Processing, Abstractive Summarization, Named Entity Recognition, Machine Learning, ROUGE**

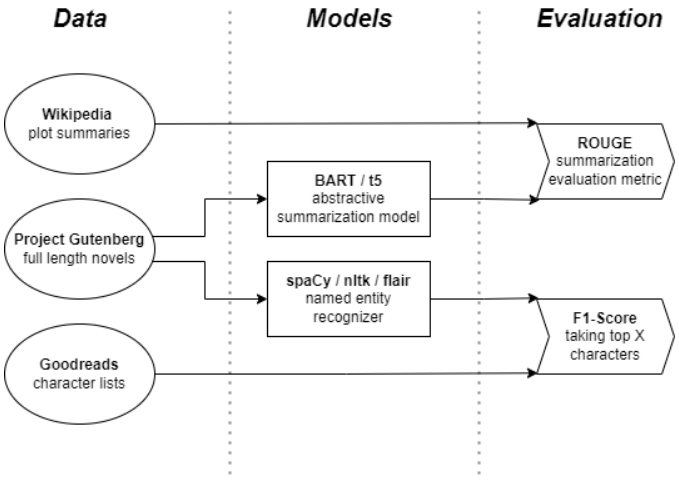

Fig. 1. General diagram of architecture

I. Introduction

Traditional book publishing is requires many resources. During the long, expensive process of book editing, marketing, printing, and publishing, writing a blurb to describe the content of the book is a relatively trivial task. As long-form writing trends away from traditional publishing and towards much more streamlined online publishing, the creation of a blurb, or short summary, is often left behind.

To address this growing collection of digital long-form fiction without summaries, Blurble investigates the use of abstractive summarization models in fiction books. To give further information on a book's content, we also investigate the use of Named Entity Recognition in identifying and listing Main Characters.

Abstractive summarization describes the task of automatically generating a concise summary from a source text. It differs from extractive summarization in that it writes new sentences instead of just extracting and copying the most significant sentences in the text.

In recent years there has been substantial research into abstractive summarization. However, most of this research is centred around summarizing news articles. The writing in news articles attempts to be clear and factual, and journalists often summarize the main points of articles into headlines and sub-headlines. These features make articles an ideal target for summarization research. Undertaking this task with novels introduces two main challenges. The first concerns the text's length; it is evidently simpler to concisely summarize a news article than a book which may be considerably longer. The second relates to writing matter and manner, the challenge of synthesizing fictional prose rather than factual reporting.

The modern approach to abstractive summarization is to first train up a language model in an unsupervised way by feeding a huge text corpus through a deep neural network. This language model is optimized for a task by giving it a labelled training set. For abstractive summarization, this training set would consist of source texts and their associated summaries.

Named Entity Recognition (NER) is a sub-task of summarization that aims to identify and label the most common proper nouns. These may be, for example, names of people, organizations, or locations. The modern approach consists of slicing sentences in a large annotated corpus, identifying each part of the sentences that relates to other parts. This massive dataset is used in the computationally expensive task of training a deep neural network to create a NER model.

Through our research, we experiment with existing models modified for long-form fictional text,as seen in fig 1 which illustrates the general architecture. We find promising results, especially with the BART-LARGE-XSUM-SAMSUM model. We identify the reliability, or lack thereof, of ROUGE evaluation as the limiting factor for improving the task.

II. Background

A. Related Works

1) Early Work: Natural Language Processing (NLP) is a field in computer science that mixes linguistics or language study with artificial intelligence. NLP researchers' primary focus is the development of technology/programs which understand human vocabulary.

Within NLP, there are many sub-fields. Automatic summarization concerns the fields of text processing and synthesis. While humans are adept at reading comprehension, reading and summarizing a single text accurately, it can take a significant amount of time, let alone a whole corpus. Summarizer models are advantageous as they can automatically detect vital information within a large given text and present them back quickly and in logical order. These models efficiently summarise millions of words in a relatively short time but come with various limitations. Said limitations often involve accuracy and readability.

There have been many summarization approaches and models over the years, the first of which was created by Hans P Luhn in 1958, where he used word frequencies to determine the importance of words or sentences to create an 'auto-abstracter' [1]. Due to high word frequencies, the common stop-words were taken out in Luhns' 'auto-abstracter'. However, this is rudimentary logic, and more enhanced methods exist to achieve these techniques.

H.P.Edmundson quickly noticed that word frequency was not enough and that not all sentences were equally weighted. In 1969 Edmundson proposed three additional methods to improve automatic extraction [2]. To better weigh the features, these included the acknowledgement of sentence location (sentences under specific sections are more important), cue words (words near pragmatic phrases like 'impossible' are affected and weighted, respectively) and heading words (titles and heading are desirable). The main limitation for both Luhn and Edmundson was the small computer storage capacity and the expensive procedure to achieve these results.

An abrupt change in how researchers developed Summarizer models arose when Gerald DeJong, in 1979, developed FRUMP (Fast Reading Understanding and Memory Program), which uses 'sketchy script' that scan, highlight and store essential information [3]. The scripts were tailored to specific sections and were used as templates to generate summaries. While innovative, the 'sketchy scripts' had to be made manually for each domain. It was time-consuming and contradicted its automation objective by introducing a manually created template.

During the 1990s, there was plentiful development. However, the research stayed on the same path of empirical and rational approaches allowing for slight improvements within the summarization field but no breakthroughs. In 2004 Kaikhah, K used Neural Networks (NN) to summarize news articles [4]. The model consisted of training the NN on what the output should look like with manual intervention to ensure it included or excluded the correct sentences within the summary. It also used feature fusion which consisted of pruning the neural connections with small weights and enhancing sections with trending relationships, as well as sentence selection which was also used back in Edmund's 1969 research paper.

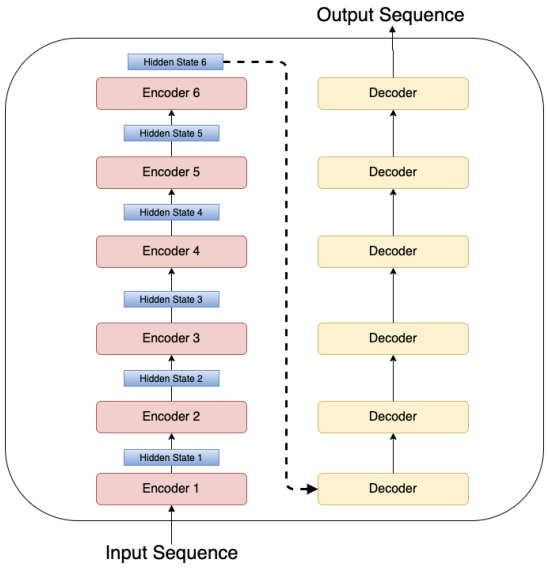

Fig. 2. 6 layered Translation Encoder decoder simplified architecture [7]

.

Named entity recognition (NER) was hypothesized in the 1990s and developed in the late 2000s. Early work in 1995 was introduced in the MUC-6 or the 6th Message Understanding Conference [5]. Its primary purpose involved recognizing entities (place or location) so that it would be easier to search for when annotating text, no matter what language. More recent work in named entity recognition involves automatic identification and tagging of these entities using machine learning; it can be contextualized and extracted for use, e.g. Identifying cities in an article and outputting the top 3 cities mentioned [6].

2) Recent Work: Then, breezing to the 2010s, machine learning techniques drastically improved and became standard. These revolutionized the Summarizer models using deep learning transformers and increasing performance with pre-trained models. GPT or Generative Pre-trained Transformer model is the first to introduce pre-training rather than having models trained on task-specific areas. It used unsupervised pre-training but supervised fine-tuning. It is trained on a large book corpus to help the models 'learn' the language before fine-tuning the model to a labelled dataset for a specific target task. The task of summarization is fine-tuned with the CNN/Dailymail dataset of articles and their corresponding summaries. GPT's architecture consists of 12 layered decoder where the transformer has a self-attention mechanism to train and help train the models; an example of a simplified 6 layer encoder is seen in figure 2 [8]. Its successor, GPT-2, was trained on an even larger corpus of data, had 96 attention heads compared to the 12 from GPT-1, and the

size of word embeddings went up ninefold [9]. With GPT-3, these numbers skyrocketed once again [10] however, when it came to summarization, these models failed to or barely outperformed existing text summarization models.

After GPT introduced these methods, even if they were not entirely successful in the summarization field, it allowed other models to incorporate pre-trained transformers to be optimized for summarizing. BERT standing for Bidirectional Encoder Representations from Transformers, uses similar approaches to pre-training but only has an encoder since, for text generation, the decoder is not needed in the system [11]. A fine-tuned BERT model outperformed most summarization systems in 2018. BERT reached an average ROUGE F1 score of 43.25 on extractive and 41.72 on abstractive summarization when creating summaries for the DailyMail/CNN test sets.

BART, or Bidirectional Auto-Regressive Transformer, is the main focus of our research. It has recently been showing great results within the abstractive summarization field, receiving F1 scores of 44.16 on the CNN/Dailmail and 45.14 on the XSUM test sets using the base-BART model[12]. BART is unique since it uses a bidirectional encoder, like BERT, for seq2seq and a left-to-right decoder, like GPT-2, for the architecture. The architecture will be explained in more detail in the theory section. Similar to previous models, it is pre-trained on massive data sets. This allows the model to better understand the language before generating text and has many fine-tuned options, which is beneficial when acquiring optimal abstractive summaries [13].

There is a common theme within most papers, almost all of which target shorter texts and news articles in particular, which means that there is a large gap between how these models and their fine-tuning work on long books with over 200 thousand words compared to shorter articles of 200 words. Furthermore, after the summarization is complete, there are no clear indicators of who or what the main characters are. Hence, implementing a named entity recognizer to showcase character significance can benefit the summary reader.

Some recent work does focus on longer text, but lack key features. One of which includes new methods of using a modified version of Hidden Markov Models (HMM) [14]. While the results are interesting, it focuses on extractive summarization with dated models. In a different paper, the use of a long sequence summarizer is explored to generate Wikipedia entries from their respective works cited [15]. Their abstractive model uses a decoder-only architecture, a modified version of a Transformer Decoder model [16] called the Transformer Decoder with Memory-Compressed Attention (T-DMCA). This model shows excellent results. However, the main limitation is the max-tested training input length of 11000 words and decreased performance with increased input length .

Experimenting with existing summarization models for larger text can give insight into how effective the systems are when pushed to their limits. In addition, exploring the different fine-tuning and which datasets provide better feedback when dealing with books can also help establish new and unique ways to improve the models for long-text inputs.

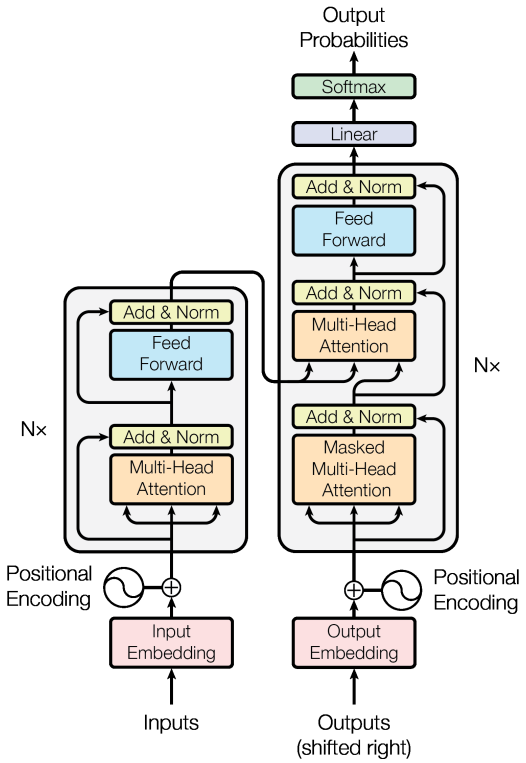

Fig. 3. The Transformer - model architecture [17]

.

B. Theory

There are two main methods of summarization. These include an extractive method that groups and takes important or frequently repeated sentences or sections that produce a subsection from the original book or article, essentially copying key sections from the text. The other method, abstractive summarization, generates a summary that includes new phrases. It tries to generate essential information based on the text rather than only reiterating what has already been written[18]. Abstractive summaries are the main focus when working with BART; however, some experimentation with extractive models was included in the results[18].

Most of the recent models use an encoder-decoder transformer. BART also uses this method, and the transformer model architecture is seen in figure 3, where the encoder takes in a sequence of symbol representation inputs and encodes it to become a sequence of continuous representations—followed by the decoder that produces an output for each symbol representation at a time. As this repeats, the transformer consumes the previous outputs as additional inputs for future text generation. There are attention mechanisms within the encoder and decoder to smoothly and efficiently transfer input information. The Large BART model also offers twice as many encoders and decoders within the transformer [17].

Attention mechanisms are used for long-distance interactivity in NLP tasks, first introduced when the encoder had limited

access to the information being provided from the input. These mechanisms are used to create shorter and more efficient ways to connect relevant input sections in a flexible approach. Using a combination of relevant encoders and weights can attend to the model's different desirable parts. As seen in figure 3, there are multi-head attention mechanisms inside of it; this allows many scaled dot product attention functions with dimensionality d to pass through them. This scaled dot product is divided by dimension d, and a softmax function is applied to obtain the weights [17].

However, before running through the linear and softmax function inside the sublayers, it goes through feedforward networks, which run linear transformations using the ReLU activation function, allowing the dimensionality of the inputs and outputs to be the same at each layer. [17].

NER has gone through various approaches before the modern approach of deep learning. In its early days, NLP achieved good performance in domain-specific applications through human engineering. This often works with a dictionary-based approach where key terminology is already identified. On top of the human engineering cost, an additional shortcoming of this approach is that it works best when the lexicon is exhaustive.

NER can also be done through unsupervised or supervised learning approaches. Unsupervised learning is often done through clustering-based NER. The idea is to identify named entities based on similar vocabulary contexts. In supervised learning approaches, words in a training corpus are labelled depending on the type of named entity they are (person, location, organization). Feature engineering is vital to this approach and can include character level, word level, list lookup, and document level features [19].

III. Method

A. Data

The task requires full novel texts, with their associated summaries & lists of characters. To obtain a relatively large and consistent set of novel summaries, we turn to Wikipedia [20]. Using the MediaWiki API [21] we can extract the "Plot Summary" section of a book's Wikipedia page, given that the page and section both exist. Similarly, lists of characters can be scraped directly off the book's Goodreads page [22].

Acquiring full-length novels requires finding well known books that have complete Wikipedia and Goodreads pages while being freely available under Intellectual Property law. Under the United States, copyright law, works published before 1978 remain under protection for 95 years after their publication [23]. This means works published before 1927 are in the public domain and freely available. The Project Gutenberg digital library gathers such works in e-book format [24]. These can be found and downloaded using the Gutenberg API [25]. The texts are then cleaned up in an attempt to remove all preamble and appendices.

Unfortunately, our approach does not yield enough reliable book-summary pairs to train or fine-tune a model. For our objective, to test and evaluate the performance of different models, we focus on having a high-quality dataset rather than one of high quantity. We hypothesize that user-generated Wikipedia summaries will be of higher quality the more popular the book is, based on the assumption that more user interest leads to more and better user edits. *thegreatestbooks.org* algorithmically generates a list of "best books" based on 130 "best books" lists from various sources [26]. Using this list as a heuristic for most popular books and filtering it for books published before 1927, we end up with a list of 689 book titles.

This list must be further filtered to include books we can use solely. Many books are non-fiction or in eclectic formats. Most Wikipedia summaries are too short or focus on cultural surroundings and impact rather than the plot itself. A lot of Wikipedia pages do not even include a plot summary section. Also filtered are books over 200k words and summaries less than 100 words. Books where the text is over a thousand times the length of the summary or under 20 times the length are also removed. After these filtering steps, we have left a dataset of 106 book-summary pairs.

The dataset for Main Character Identification is generated from the same list of book titles. Books are filtered on the 1 million character limit of some of our NER models. If a Goodreads character list can be found, it is excluded if it contains less than three characters or more than 50. This leaves us with 101 books and associated Goodreads character lists.

One main challenge of novel summarization is the great length disparity between the novel text and its summary. The CNN/Daily Mail dataset BART is often fine-tuned on articles averaging 766 words, with summaries averaging 53 words. This dataset has a 6.9% Summary-to-text length ratio [27].

	Novel WC	Summary WC	Character List Length
Mean	76,788.72	683.79	10.23
Median	69,577.5	673	8
σ	46,588.42	380.83	7.07

TABLE I
WORD COUNT FOR GUTENBERG NOVELS, AND WIKIPEDIA SUMMARIES. NUMBER OF CHARACTERS OF GOODREADS CHARACTER LISTS

Meanwhile, as shown in table I, the novels in our dataset average 76,789 words, while our Wikipedia summaries average 684 words, these synthesize the novels down to 0.9% of their original length. In order to generate summaries six times more compact than those BART is trained on, we resorted to novel approaches.

B. pre-processing

The pre-processing methods are either implemented within BART or manually added to enhance the model further. As seen in the subsections below, the methods used for pre-processing have some general techniques like tokenization, which almost all NLP programs would use, and some niche methods like text chunking.

1) Chunking: Our approach to summarizing full length books consists of splitting the book into sections, each of which are individually summarized. Chunking is a way of

splitting up large files into scaled segments. First by replacing all full stops, exclamation marks and question marks with the end of sentence tags but keeping their respective punctuation to keep the format. These tags allow easier splitting within the model. Setting a limit of upwards of 1024, but an average of 650 was used within our model. The chunk size allows our models to process lengths of text they are optimized for. Creating a for-loop to append words within sentences ensures that the current chunk does not exceed that limit; if it does, it splits, stores, and starts on a new chunk.

2) Tokenization: Tokenization is an essential step in any NLP task. It separates a given text into segments. However, unlike chunking, it focuses on smaller tokens, like words or sub-words. For example, a minor sentence is broken into a segment of tokens using spaces to indicate each token. Having the text in an array of tokens helps the NLP models accept the information smoothly [28]. Tokenization technique selection is available within BART; however, the default RobertaTokenizer was selected after testing. Most of its parameters can also be tweaked. The other technique was the RobertaTokenizerFast which would impede result accuracy to save on computational time [29].

3) Normalization: Normalization unifies the entire text making it simple to work with. Replacing characters, lower casing the text, removing duplicates, white space, and diacritics all enhance the performance of the models [30].

C. Models

1) Pegasus: Pre-training with Extracted Gap-sentences for Abstractive SUmmarization Sequence-to-sequence, also known as PEGASUS, is a model solely used for abstractive summarization; its pre-training is the generation of gap sentences via self-supervision. A similar method is used in BART, which is discussed in the upcoming pretraining section [31].

2) DistilBERT: BERT standing for Bidirectional Encoder Representations from Transformers, is the base version of DistilBERT. DistilBERT is a smaller, cheaper and faster version of BERT. Reduction in all aspects, including computational time, while preserving over 95% of its performance. Pre-trained on masking and sentence prediction. [32]

3) T5: T5 is a model that takes the text in the input and also outputs text (Text to Text Transfer Transformer) using both supervised and unsupervised methods for pre-training. Using tasks such as reading comprehension, classification, translation, and summarization to learn [31].

4) BART: BART is a seq to seq transformer model based on the encoder-decoder methods, where the inputs and outputs are for a natural language or, in this case, sentences and summaries. At the same time, the encoder takes in the high dimensions of the input, which is later sent to the output and mentioned in the background model section on transformers. The BART model is pre-trained on a colossal corpus which includes popular sites like Wikipedia and Book-Corpus. BART's large and varied pre-training corpus, and its broad range of fine-tuning applications and model types make it a versatile, and state-of-the-art model [13].

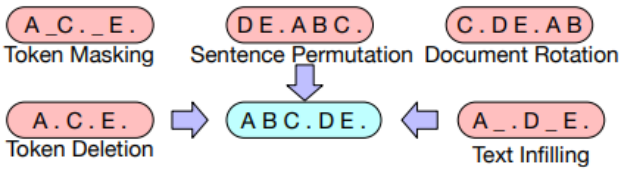

Fig. 4. Transformations visualized [13]

In our research, the base BART model was the one we planned on using. However, upon further evaluation of different fine-tuned BART models on hugging face [33], which included BART-Large, and fine tuning on CNN, XSUM SAMSUM, distbart, MNLI, datasets. In the theory section, the large BART model was chosen over the base since it offered twice as many encoders and decoders along with larger indexing. We eventually selected a version of BART-LARGE fine-tuned on the CNN/DAILY MAIL training sets, as well as one fine-tuned on XSUM SAMSUM. These are explored in the fine-tunning section.

D. pre training

Pre-training is essential when working with large data on a small target task. Which fits our task of summarization of large books to small summaries of around 1% of the original book. In addition, pre-training has been shown to significantly improve robustness and minimize uncertainties in models. [13]. Having the models pre-trained on broader data, followed by fine-tunning, drastically improves results, especially in the summarization field [34].

There are different pre-training methods, which would involve transfer learning and supervised pre-training along with self-supervised learning and self-supervised pre-training. Pre-training is essentially training the model to 'understand' the language rather than have a specific task to solve. This learning can either be simple, e.g. understanding spellings, or very complex, where the program understands that a cruise ship cannot fit in a purse. This knowledge can eliminate a lot of obscure or innacurate abstractive summarizations results.

BART's unsupervised pre-training is done by taking input documents and transforming or corrupting them. The model trains with the objective of restoring these documents to something matching their original state. By inputting randomly distorted text and 'fixing it', the model can take any unlabeled document and train from it. In extreme cases where most of the document information is removed or filtered, BART starts to implement and fill in those gaps, optimizing for text generation. The model's indication of success is based on the restoration compared to the original document. Transformation methods include Token masking, Token Deletion, Text Infilling, Sentence Permutation and Document Rotation, as seen in a simplified way in figure 4 [13].

E. fine-tunning

BART has various applications; however, when explicitly working with summarization, it needs to be fine-tuned on sequence generation tasks. Due to BARTs' autoregressive decoder, it can directly be fine-tuned for specific tasks. The input information is manipulated. This is a similar method to the aforementioned pre-training. The encoder is the input, and the decoder is the output, where it generates autoregressive results.

There are many tasks or finetuning datasets, including CNN/Daily Mail, a news article summarization dataset. It consists of pairs of short articles found on the CNN/Daily Mail websites, and short human written summaries for said articles. [35].

XSUM is also based on news articles, this time from the BBC website. The summaries found in this dataset are much shorter, making this training set very useful for tasks where text must be summarized into relatively small summaries. [36].

Samsum dataset contains thousands of text message conversations, written by linguists to simulate every day texting dialogue. The summaries within this dataset are written by the very same linguists. It is also one of the finetuned datasets chosen for conclusive outputs. [37].

One of the finetuned BART datasets chosen was the XSUM-SAMSUM set. This means the model was finetuned twice, first on the XSUM abstractive news summaries, followed by Samsum filled with messages and dialogue. Initially, this combination does not seem ideal when working with extensive stories. However, the expected desired result is that this model will be optimal for both generating relatively shorter summaries and summarising book dialogue.

1) Named entity Recogniser: Using Named entity recognition can help aid summarization systems in directly identifying the main components of an important object or tag.

Named entity recognition seeks to identify and tag each entity within a given text. NER uses IOB tagging where I stands for inside, O, for outside and B for beginning(e.g. sentence starting with Ireland tagged with B-Geo). The tags help identify the label, which helps hone in on the specific tags needed. Specifically, the tag ' PERSON' is used to identify characters within a story. [6]

The three models used are:

- **NLTK**, a standard python NLP library. The named entity recognizer is a supervised learning algorithm trained on a corpus. Again, as mentioned previously, it learns to tag entities and specifies their type where we are only interested in 'PERSON' [38].
- **Spacy**, almost a direct upgrade to NLTK; it uses deep learning to tag and encode text. All the typical NLP pre-processing is done within the model when it takes a text input. When the input is added, the words get encoded into vectors where a convolution neural network can contextualize and work with the data. CNN is the main driving factor in the POS tagging and named entity recognizer within SPACY [39].
- **Flair**'s NLP modules also consist of POS and sentiment along with others, which are also very similar to the other two; however, all of the models are trained in various ways with separate corpora, which ultimately leads to different results, as seen in the result section.

F. Evaluation

1) ROUGE: ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is the industry standard evaluation metric for the task of document summarization [40]. It counts the overlapping content between a machine-generated summary and a sample "ideal" human-generated summary. To calculate the accuracy of a result, ROUGE is based on an F1 metric. This metric assigns a score after taking precision and recall into account .

Precision is a measure of quality that computes the percentage of correct generated values. In ROUGE, it is calculated with:

$$\frac{amount\,of\,overlapping\,content}{amount\,of\,content\,in\,machine\,generated\,summary}$$

The recall is a measure of quantity which computes the percentage of correct values found by the generated results. In ROUGE, it is calculated with:

$$\frac{amount\,of\,overlapping\,content}{amount\,of\,content\,in\,ideal\,summary}$$

We can then calculate the F1 score using:

$$2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Different ROUGE measures employ different definitions for "content". ROUGE-N measures n-gram overlap. We will use ROUGE-1, which measures the overlap of unigrams or single words, and ROUGE-2, which measures the overlap of bigrams, or consecutive pairs of words. ROUGE-1 lets us know if the words used in summary are relevant. On top of this, ROUGE-2 considers word order and word pairs.

ROUGE-L uses the Longest Common Subsequence (LCS) between the generated and ideal summary. LCS finds the most extended sequence of words in the same order but not necessarily consecutive. Using this sequence as "overlapping content" in our metric allows for more flexibility since it intrinsically incorporates the longest matching n-grams without defining a fixed n-gram length.

In simple terms, ROUGE-1 measures if the same individual words are being used in both summaries, while ROUGE-2 and ROUGE-L compare longer sections of text to try and evaluate if the words form sentences with similar meanings.

Ideally we want our generated summary to match our target summary in length. If it is too long, it will result in a low precision score. If it is too short, it will result in a low recall score.

2) Naive Baselines: Since we created our dataset, we do not have other papers to compare our results. We must therefore generate some naive summarization baselines to evaluate our results against.

A typical summary baseline in the field is extracting the maximum sentences in the documents. In our longer texts, we take the sentences in the first 1000 words of the books to somewhat match the average Wikipedia summary length.

This 'leading sentences' approach is beneficial in news articles which try to encapsulate the main points at the start. However, novels might not present their main plot points until much later in the text. Therefore, to naively capture events from all points in the book, our Random1000 baseline extracts sentences at random locations to form a 1000-word summary.

With a similar goal in mind, our chap-starts and chap-ends baselines take the first few or last few sentences of each chapter to create a 40-sentence summary.

3) Main Character Recognition Evaluation: In order to correctly measure the precision and recall of our Character Recognizer, we need to go further than a simple F1 measure. The difficulty is that our Goodreads test-set is user generated, and each entry lists an arbitrary number of main characters. Meanwhile, our Character Recognizer attempts to identify all characters and ranks them in order of several occurrences, which generates a very long list. For our evaluation, we must trim this list by only keeping the highest occurring characters.

Picking a fixed length for our model-generated list is troublesome. Let *N_test* be the number of characters in the Goodreads list, and *N_model* the number of highest occurring characters we are picking from our model-generated list.

When evaluating against a concise test list, such that *N_model>>N_test*, there will be a lot of false positives, and so the precision score will be meagre. Evaluating against a much longer test list, such that *N_model<<N_test*, will result in many false negatives, leading to very low precision. Matching *N_model* with *N_test* for the respective test list is not a bad solution, but still disregards characters ranked further down than *N_test* in our generated list.

Our solution seen in Fig 5 is to first take the top 2**N_test* characters in our generated list. We then trim the list at the last correctly identified character and calculate the F1 metric on this trimmed list.

We test our generated results with three degrees of strictness:

- **Any_Match** counts a found character as valid if part of either their first or last name is found in the test list.
- **All_Match** individually weighs the matching of first and last names.
- **Strict_Match** requires both the characters' first and last names to match for it to be correct.

IV. Results and Discussion

A. Summarization Results

The results displayed in table II show the BART and BERT models leading in terms of performance. While all models outperform the baselines in ROUGE-2 and ROUGE-L metrics,

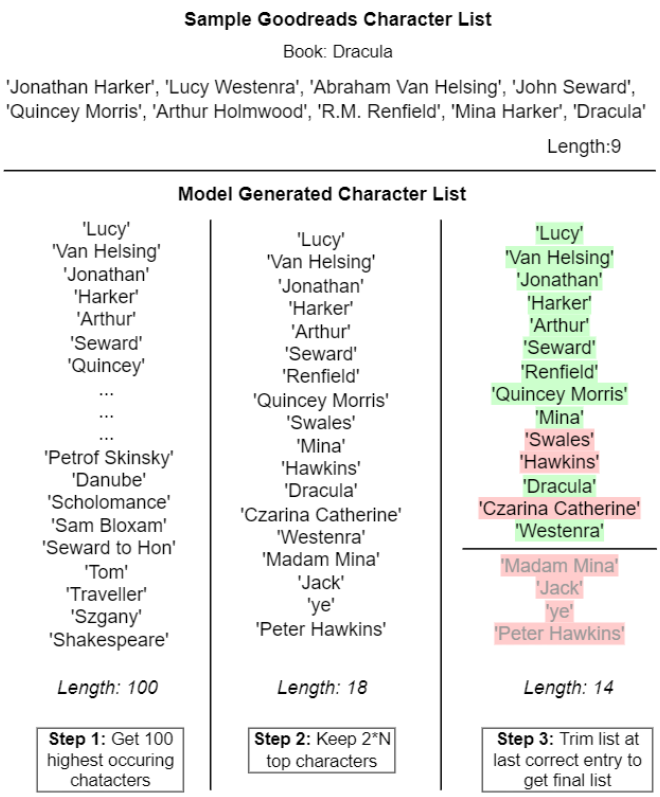

Fig. 5. Character list generation process

Method	ROUGE-1	ROUGE-2	ROUGE-L
BART Large XSUM SAMSUM	**37.76**	**9.16**	**16.22**
Pegaus XSUM	30.01	7.56	13.82
T5	33.27	6.66	14.14
BART Large CNN/Dailymail	36.07	7.76	14.97
DistilBERT	35.86	8.01	15.20
first1000	33.49	5.37	13.30
random1000	35.14	5.22	13.00
chap-starts	31.67	5.52	13.16
chap-ends	33.41	5.68	13.66

TABLE II
SUMMARIZATION MODELS COMPARED TO BASELINES

which measure sentence meanings, the Pegasus and T5 models under-perform in ROUGE-1, which measures individual word use. The BART Large XSUM SAMSUM, which is fine-tuned for both shorter summarization, as well as summarizing conversations.

An initial hypothesis also included that the T5 model would outperform BART due to its larger training corpus. However, due to its lack of fine-tuning models and its smaller indexing size of 512 (affecting chunk size), we see a much lower ROUGE metric.

The correlation between word counts and summarization performance can be seen in fig 7. We can see that while our model-generated summaries reflect the length of their original texts, the length of Wikipedia user-generated summaries are completely independent from that of their respective books. Looking at ROUGE-1 correlations, we see that longer books,

Wikipedia Summary Extract	BART LARGE XSUM SAMSUM Sumary Extract
• • •	• • •
When Dorothy and her friends meet the Wizard again, Toto tips over a screen in a corner of the throne room that reveals the Wizard, who sadly explains he is a humbug—an ordinary old man who, by a hot air balloon, came to Oz long ago from Omaha. He provides the Scarecrow with a head full of bran, pins, and needles ("a lot of bran-new brains"), the Tin Woodman with a silk heart stuffed with sawdust, and the Lion a potion of "courage". Their faith in his power gives these items a focus for their desires. He decides to take Dorothy and Toto home and then go back to Omaha in his balloon. At the send-off, he appoints the Scarecrow to rule in his stead, which he agrees to do after helping Dorothy return to Kansas. Toto chases a kitten in the crowd and Dorothy goes after him, but the ropes holding the balloon break and the Wizard floats away.....	Dorothy and her friends are in the Throne Room of the Great Oz. Oz promised to send Dorothy back to Kansas when the Wicked Witch was destroyed. Oz fooled the Scarecrow, Dorothy, Toto and the Tin Woodman into thinking he was a wizard. He is a ventriloquist and a balloonist. He was born in Omaha, Kansas. The Emerald City was built by Oz many years ago. Oz has been good to the people, and they like him, but ever since the Palace was built he has shut himself up and would not see any of them. Oz is a bad Wizard, but he can't keep his promises. Dorothy wants to go back to Kansas. Oz wants her to stay with him for two or three days. The Tin Woodman wants to take the Scarecrow\'s head off. The Scarecrow wants to get his brains back. Oz made a heart out of silk and stuffed with sawdust. He put a patch on the Tin Woodman\'s breast. He made a drink out of a green bottle and a green-gold dish. He gave it to the Cowardly Lion, who drank it. Oz is going to cross the desert in a hot air balloon to visit his brother Wizard in the clouds. Dorothy is going with him. Dorothy wanted to go with Oz in the hot air balloon, but the ropes broke and the balloon went into the air without her. Toto had run into the crowd, so Dorothy picked him up and ran to the balloon.....
• • •	• • •

Fig. 6. Extracts taken from the summaries of *The Wonderful Wizard of Oz*. On the left, a paragraph from the Wikipedia plot summary of the book. On the right, the part of a BART-generated summary describing the same section of the story

Target Text	*"Harry goes to the market and pays off a vendor."*
Generated Text 1	*"Harry goes to the sea and kills off a vendor"*
Rouge 1	80.0
Rouge 2	55.6
Generated Text 2	*"Harry settles his debts at the bazaar with one of the vendors"*
Rouge 1	27.3
Rouge 2	0.0

TABLE III
ROUGE SCORES FOR HYPOTHETICAL SUMMARIES. EMPHASIZING SHORTCOMINGS OF ROUGE

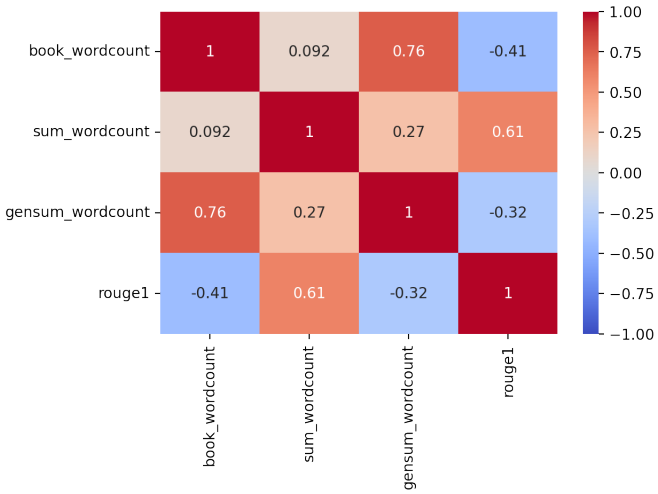

Fig. 7. Pearson's Correlation matrix of word counts against metric

with correspondingly longer generated summaries, tend to perform a little worse than shorter books. The task of concisely summarizing text, to match a pre-existing summary, is harder the longer the original text.

Evaluating summaries of long-form fictional literature is more challenging than evaluating summaries of short factual articles. The human-generated reference summaries used for training and testing a model are less valuable in the former. A news article will likely have a clear message with several essential points. A novel will likely have many plot points of varying importance. Picking which of the many plot points to include in a concise summary is a challenging task for a machine and a human. Two human summaries for the same book might not include the same plot points. An excellent machine-generated summary might be very different from the reference summary simply because it focuses on different minor plot points.

Even in the case of two summaries with the same plot points, we run into one of the classic problems of abstractive summarization. Sentences with the same meaning might use completely different words. Furthermore, when it comes to human summarization, each person will summarise based on their background and specific style, adding even more uncertainty to the summaries provided. This is reflected in our relatively low ROUGE-L scores compared to what is usually seen in abstractive summarization tasks.

This highlights some of the shortcomings of the ROUGE evaluation metrics. The metrics are useful for identifying word unit overlap, returning better results when the general subject matter is similar. However, ROUGE is unable to consider the meaning of a sentence, as seen by the examples in table III.

Future work for our summarization of novels would include training the talked-about models ourselves with a huge corpus of books rather than using a pre-trained model. Furthermore, additional exploration of minor changes to tokenization, the chunking and testing of various types of book data (50k word books, 200k books word, 1 million word books) would benefit future advancements.

B. Main Character Identification results

Model	Any Match	All Match	Strict Match
spacy NER	**51.25**	**44.41**	**16.50**
nltk NER	47.81	39.41	11.25
flair NER	48.74	40.33	14.22

TABLE IV
MAIN CHARACTER IDENTIFICATION RESULTS

As expected, our results, seen in table IV, show the newer spacy model outperforming both flair and nltk. While the results prove that this method is able to identify some main characters, it is far from reliably identifying all of them.

The main cause for error in our tests comes from individual characters being identified multiple times under different names. This happens with the first and last name being identified individually, but also when the character is called by their title or by a nickname. The next logical step for the future would be to develop a system which identifies the different names used to refer to a single character.

Another source of inaccuracy is that counting name occurrence is not a perfect heuristic for character importance. For instance, in a book written in the first person, while being the principal character, the narrator's name will rarely appear outside of dialogue. A perfect Main Character Identifier would need to take into account the importance of the character's name in the context it appears in.

With the main characters identified, another challenge would be to attempt to create character descriptions. By solely taking the sentences or sections of a book that pertain to an identified character, it would be interesting to train a model to generate a description of that character and its role in the book.

V. Conclusion

We explore the use of existing abstractive summarization models in the task of generating summaries for full length novels. These models being optimized for summarizing short factual articles, creative methods had to be employed to apply the models to longer fictional texts and generate comparitively shorter summaries. While we report results outperforming our baselines, it is clear much work remains to be done on the challenge of book summarization, particularly in improving current evaluation metrics to depend less on exact word choice.

We also explore the use of NER models in identifying a novel's main characters. These perform well enough that the generated lists of characters may come in very useful to create character descriptions or aid in the task of summarization.

References

[1] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research and Development*, vol. 2, no. 2, pp. 159–165, 1958.

[2] H. P. Edmundson, "New methods in automatic extracting," *J. ACM*, vol. 16, no. 2, p. 264–285, apr 1969. [Online]. Available: https://doi.org/10.1145/321510.321519

[3] I. DeJong, Gerald Francis, "Nskimming stories in real time: An experiment in integrated understanding,," 1979. [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA071432.pdf

[4] K. Kaikhah, "Automatic text summarization with neural networks," in *2004 2nd International IEEE Conference on 'Intelligent Systems'. Proceedings (IEEE Cat. No.04EX791)*, vol. 1, 2004, pp. 40–44 Vol.1.

[5] R. Grishman and B. Sundheim, "Message understanding conference-6: A brief history," in *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, ser. COLING '96. USA: Association for Computational Linguistics, 1996, p. 466–471. [Online]. Available: https://doi.org/10.3115/992628.992709

[6] A. Roy, "Recent trends in named entity recognition (NER)," *CoRR*, vol. abs/2101.11420, 2021. [Online]. Available: https://arxiv.org/abs/2101.11420

[7] M. Deshpande, "The transformer: A quick run through," May 2020. [Online]. Available: https://towardsdatascience.com/the-transformer-a-quick-run-through-ce9b21b4f3ed

[8] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[10] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[11] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," 2019. [Online]. Available: https://arxiv.org/abs/1908.08345

[12] Z. Zhao, S. B. Cohen, and B. Webber, "Reducing quantity hallucinations in abstractive summarization," *arXiv preprint arXiv:2009.13312*, 2020.

[13] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *CoRR*, vol. abs/1910.13461, 2019. [Online]. Available: http://arxiv.org/abs/1910.13461

[14] D. Bamman and N. A. Smith, "New alignment methods for discriminative book summarization," *arXiv preprint arXiv:1305.1319*, 2013.

[15] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," *arXiv preprint arXiv:1801.10198*, 2018.

[16] Z. Yang, C. Zhu, R. Gmyr, M. Zeng, X. Huang, and E. Darve, "Ted: A pretrained unsupervised summarization model with theme modeling and denoising," 2020. [Online]. Available: https://arxiv.org/abs/2001.00725

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[18] M. A. Pai, "Text summarizer using abstractive and extractive method," 2014. [Online]. Available: https://www.ijert.org/text-summarizer-using-abstractive-and-extractive-method

[19] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *CoRR*, vol. abs/1812.09449, 2018. [Online]. Available: http://arxiv.org/abs/1812.09449

[20] Wikipedia contributors, "Wikipedia," 2022, [Online; accessed 6-July-2022]. [Online]. Available: https://en.wikipedia.org/

[21] J. Goldsmith, "Wikipedia," https://github.com/goldsmith/Wikipedia, 2013.

[22] G. Inc., "goodreads," 2007. [Online]. Available: https://www.goodreads.com/

[23] Copyright Law of the United States. 17 U.S.C. § 301-305.

[24] M. Hart, "Project gutenberg," 1971. [Online]. Available: https://www.gutenberg.org/

[25] R. Angelescu, "Gutenbergpy," https://github.com/raduangelescu/gutenbergpy, 2016.

[26] S. Sherman, "The greatest books: The best books," 2009. [Online]. Available: https://thegreatestbooks.org/

[27] R. Nallapati, B. Zhou, C. dos Santos, Ç. Gulçehre, and B. 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. [Online]. Available: https://aclanthology.org/K16-1028

[28] J. Webster and C. Kit, "Tokenization as the initial phase in nlp," 01 1992, pp. 1106–1110.

[29] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[30] R. Elbarougy, G. Behery, and A. El Khatib, "The impact of stop words processing for improving extractive graph-based arabic text summarization," *International Journal of Scientific & Technology Research*, vol. 8, pp. 2134–2139, 11 2019.

[31] T. R. Goodwin, M. E. Savery, and D. Demner-Fushman, "Flight of the pegasus? comparing transformers on few-shot and zero-shot multi-document abstractive summarization," in *Proceedings of COLING. International Conference on Computational Linguistics*, vol. 2020. NIH Public Access, 2020, p. 5640.

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[33] huggingface. [Online]. Available: https://huggingface.co/models

[34] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang *et al.*, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, 2021.

[35] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015.

[36] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization," 2018. [Online]. Available: https://arxiv.org/abs/1808.08745

[37] B. Gliwa, I. Mochol, M. Biesek, and A. Wawer, "SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization," in *Proceedings of the 2nd Workshop on New Frontiers in Summarization*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 70–79. [Online]. Available: https://www.aclweb.org/anthology/D19-5409

[38] R. Jiang, R. E. Banchs, and H. Li, "Evaluating and combining name entity recognition systems," in *Proceedings of the Sixth Named Entity Workshop*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 21–27. [Online]. Available: https://aclanthology.org/W16-2703

[39] E. Partalidou, E. Spyromitros-Xioufis, S. Doropoulos, S. Vologiannidis, and K. I. Diamantaras, "Design and implementation of an open source greek pos tagger and entity recognizer using spacy," in *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2019, pp. 337–341.

[40] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

APPENDIX

The workload was distributed evenly throughout the project and while writing the paper. We both also helped each other when working on individual components when needed. While doing the project, the main data mining and dataset creations were done by Theo. Albert's primary focus was the model selection/creation and result generation. Along with both of us working on the Named Entity Recognition.

Throughout the writing of the paper, the background and most of the methods were written by Albert, while Theo mostly wrote the Introduction and evaluation. However, we both simultaneously did work within the conclusion results and methods.

Examples of generated results can be found on the gitlab