Ponnappa Appanna MACHIMANDA
Uranie JEAN-LOUIS

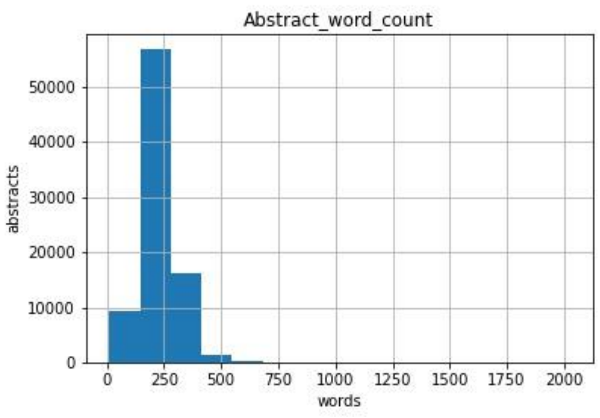
NLP PROJECT-TEXT SUMMARIZATION

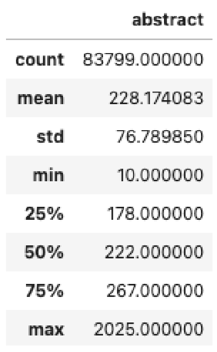
INTRODUCTION

Everyday, there are thousands of new text documents on various subjects being uploaded on the internet. These detailed documents are accessible to anyone who can use a smartphone or computer. With an abundance of resources and choice, comes the issue of selection. How can one identify what a particular document is about without spending a long time trying to read through the text?
In an increasingly busy world, time is a precious commodity. Particularly, individuals who want to leverage a diversity of opinions and perspectives in his study, must spend a lot of time investigating these various publications. However not all of the documents may have relevant material for your study.
Writing summaries is a solution to these problems. It tries to capture the essence of the entire document so that one can make a selection. However, this is a skill and not everyone is capable of creating compelling summaries which truly reflect the main content of the documents. Over the past few years, with the availability of sophisticated algorithms and machine learning models, summary making has now become a task best reserved for the machine.
In this report, we will investigate several techniques for computer generated text summarization and compare their methods and outputs. By the end of this report, we hope to convey an understanding about the most current technologies and the logic and methods used for such a task. We hope you can confidently outsource your text summarization tasks to a computer model of your choice.

MATERIALS AND METHODS

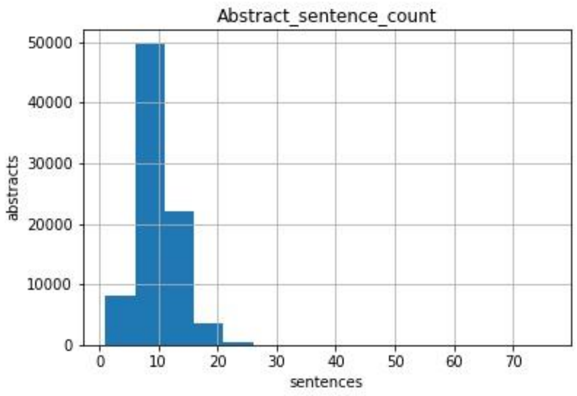
Let us start by describing the task at hand. We have been given a dataset containing thousands of abstracts from agricultural publications. The main column we will consider is the "abstracts" column. There are a total of 83,799 abstracts.

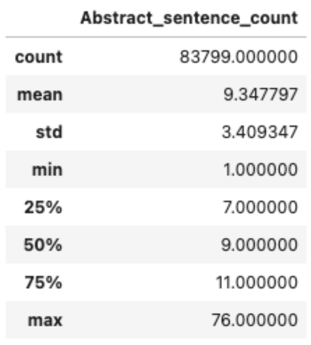




	abstract
count	83799.000000
mean	228.174083
std	76.789850
min	10.000000
25%	178.000000
50%	222.000000
75%	267.000000
max	2025.000000

75% of the abstracts are within 270 words long. The longest abstract is 2025 words long and the shortest abstract is only 10 words long.





	Abstract_sentence_count
count	83799.000000
mean	9.347797
std	3.409347
min	1.000000
25%	7.000000
50%	9.000000
75%	11.000000
max	76.000000

75% of the sentences are within 11 sentences long. The longest abstract has 76 sentences and the shortest abstract has only 1 sentence.

EXPERIMENTATION PROTOCOL

Computer generated text summarizations can be broadly divided into 2 types:

- Extractive summarization - Here the most relevant sentences are selected and generated verbatim from the main abstract. Here we do not require machine learning processes.
- Abstractive summarization – In this method, the summary is generated in natural language, using new words and sentence structures. This is much more complex than the extractive process and requires machine learning models.

We will check the different methods used for each of these types and describe the processes unique to each of them. For the purposes of testing the models, the dataset was sliced into a dataframe containing only 5 rows. With this we were able to test the timing of the different steps in the program and calculate the total time involved for running it across the entire dataset.

A. <u>Extractive summarization</u>

1. <u>Text Rank Algorithm</u>

This is based on the "Page Rank" algorithm which is used to decide the importance of a website and list them accordingly in the web search results of a google search. The algorithm factors the quality and quantity of the links in a website. The "Text Rank" algorithm models this behavior by using a similar approach for words in a sentence, instead of links in a webpage (see Figure 1).

The intuition is that, sentences which are more similar to other sentences within an abstract are more likely to be followed (web links analogy) and hence assumes a higher importance in representation (higher up in the web listing).

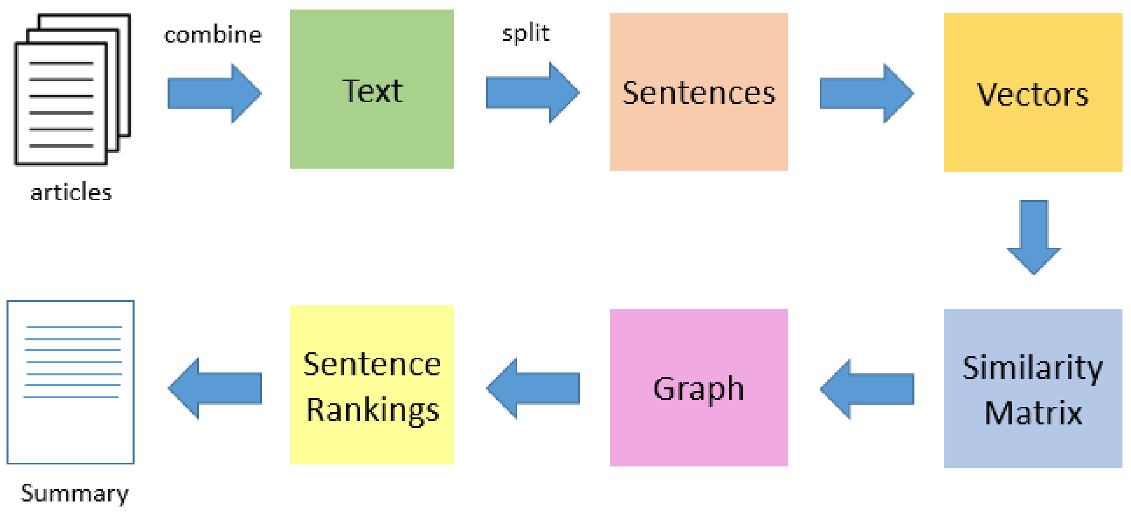


Figure 1 : Overview of TextRank method

a. *Process*

First we tokenize each sentence within an abstract (so that we can access and run operations on a sentence level).

Next we load in some predefined word embeddings. We are using the GloVe word embeddings for 100 words. This is basically a text file which has a word and its corresponding coefficient of how similar it is to the 100 meanings. There are a total of 400.000 words which have a meaning represented in terms of 100 coefficients.

Text preprocessing is done to remove any of the special characters and numbers. Further, all the characters are made into lower case. Finally, the stop words, which are the most commonly occurring words used in the English language, like: the, of, this, that etc. are removed. The words that remain would capture the main context of each of the abstracts.

Each sentence within an abstract must be represented in terms of its corresponding word embedding coefficients. If a word is not present in the word embeddings loaded, it is represented by zeroes '0'. Then a final coefficient score is calculated for the full sentence by summing each element of the 100 coeff array from each word, and dividing by the total number of words in a sentence. Now each sentence is represented by a single array of 100 elements with coeffs capturing the meaning of the sentence. This is the sentence vector for each sentence.

Then we prepare a similarity matrix which calculates the cosine similarity between sentence vectors of each abstract. So if an abstract has 6 sentences, then we have similarity matrix of shape 6*6.

Finally, we apply the page rank algorithm on this similarity matrix which then scores the sentences with an importance coefficient. These coefficients are rearranged from the highest to the lowest in importance. From this we can select the top 'n' number of sentences which captures most of the context of the entire abstract.

b. *Example:*

Let us consider the first abstract which contains 6 sentences and 198 words:

"Precision agricultural technologies (PA) such as global positioning system tools have been commercially available since the early 1990s and they are widely thought to have environmen tal and economic benefit; however, adoption studies show uneven adoption among farmers in the U.S. and Europe. This study aims to tackle a lingering puzzle regarding why some farmer s adopt precision agriculture as an approach to food production and why others do not. The s pecific objective of this study is to examine the social and biophysical determinants of farmer s' adoption of PA. This paper fills a research gap by including measurements of farmer identi ty—specifically their own conceptions of their role in the food system—as well as their percep tions of biophysical risks as these relate to the adoption of PA among a large sample of Midw estern U.S. farmers. The study has identified that farmer identity and perceptions of environm ental risk do indeed influence PA adoption and that these considerations ought to be incorpor ated into further studies of PA adoption in other jurisdictions. The findings also appear to hig hlight the social force of policy and industry efforts to frame PA as not only good for producti vity and efficiency but also as an ecologically beneficial technology."

The extractive summary defined by the top 3 sentences based on the page rank algorithm:

The specific objective of this study is to examine the social and biophysical determinants of farmers' adoption of PA.
This paper fills a research gap by including measurements of farmer identity—specifically their own conceptions of their role in the food system—as well as their perceptions of biophysical risks as these relate to the adoption of PA among a large sample of Midwestern U.S. farmers.
The study has identified that farmer identity and perceptions of environmental risk do indeed influence PA adoption and that these considerations ought to be incorporated into further studies of PA adoption in other jurisdictions.

c. *Observations:*

- As we can see, the extracted summary sentences do well to capture the context and purpose of the original abstract.
- The program can be practically executed on the full dataset using a normal laptop configuration. The longest part of the program execution is the preparation of the similarity matrix, which in our case took about 6.5 hours (for 83799 abstracts).

2. BERTSUM model

a. *Process*

Usually for extractive text summarization machine learning is not required but it could be interesting to compare one technique which used machine learning and the other does not (see above Text Rank).

The model we used make machine learning very accessible with low coding requirements. This model is called BERTSUM which is an extension of BERT but specified for text summarization.
The input of BERTSUM is different from BERT original model. First, to represent the full sentence (beginning and ending), a symbol named [CLS] token is added at the start of each sentence. This symbol separates multiple sentences and collect features of the preceding sentence (Figure 2).
Embedding is also slightly different where sentences are either categorized as Ea or Eb on whether the sentences are even or odd. For example, for document [sent1, sent2, sent3, sent4,], would assign embedding [EA, EB, EA, EB]. Then BERTSUM assigned a score to each sentence which represent how much they are relevant for the overall document. Finally, the sentences with the highest score are chosen and reordered to give the summary (Figure 2).

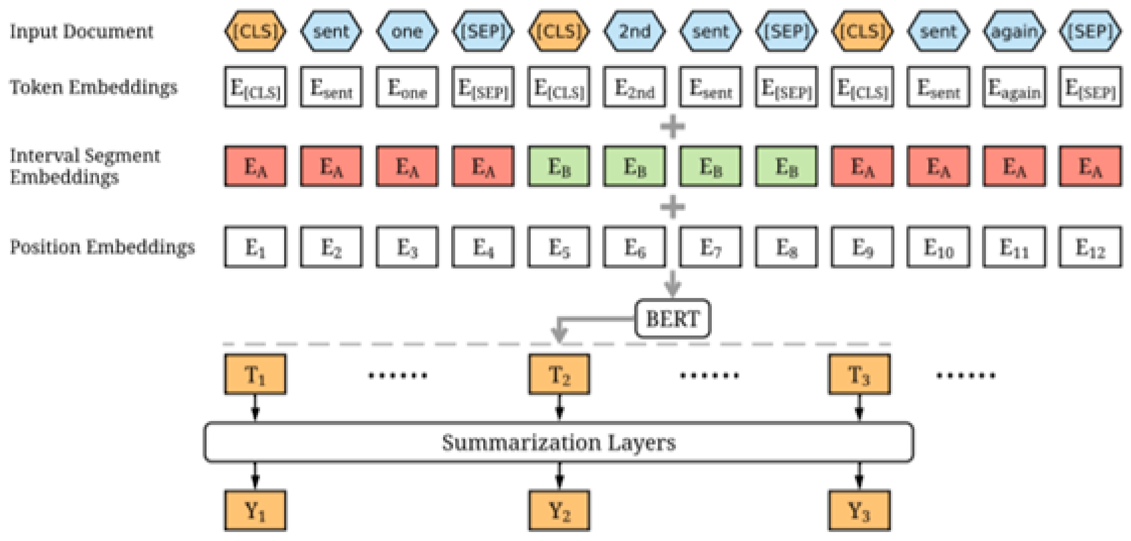

Input Document
[CLS]
sent
one
[SEP]
[CLS]
2nd
sent
[SEP]
[CLS]
sent
again
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{sent} E_{one} $E_{[SEP]}$ $E_{[CLS]}$ E_{2nd} E_{sent} $E_{[SEP]}$ $E_{[CLS]}$ E_{sent} E_{again} $E_{[SEP]}$
Interval Segment Embeddings
E_A E_A E_A E_A E_B E_B E_B E_B E_A E_A E_A E_A
Position Embeddings
E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10} E_{11} E_{12}
BERT
T_1 T_2 T_3
Summarization Layers
Y_1 Y_2 Y_3

Figure 2 : Overview of the BERTSUM model

b. *Example*

Precision agricultural technologies (PA) such as global positioning system tools have been commercially available since the early 1990s and they are widely thought to have environmental and economic benefit; however, adoption studies show uneven adoption among farmers in the U.S. and Europe. The findings also appear to highlight the social force of policy and industry efforts to frame PA as not only good for productivity and efficiency but also as an ecologically beneficial technology.

3. Comparison of the two extractives models

"Precision agricultural technologies (PA) such as global positioning system tools have been commercially available since the early 1990s and they are widely thought to have environmental and economic benefit; however, adoption studies show uneven adoption among farmers in the U.S. and Europe. This study aims to tackle a lingering puzzle regarding why some farmers adopt precision agriculture as an approach to food production and why others do not. The specific objective of this study is to examine the social and biophysical determinants of farmers' adoption of PA. This paper fills a research gap by including measurements of farmer identity—specifically their own conceptions of their role in the food system—as well as their perceptions of biophysical risks as these relate to the adoption of PA among a large sample of Midwestern U.S. farmers. The study has identified that farmer identity and perceptions of environmental risk do indeed influence PA adoption and that these considerations ought to be incorporated into further studies of PA adoption in other jurisdictions. The findings also appear to highlight the social force of policy and industry efforts to frame PA as not only good for productivity and efficiency but also as an ecologically beneficial technology."

In red the summary of BERTSUM and in blue the one of Text rank. The two models generate two different summary with indeed not the same sentences extracted from the original abstract.

B. **Abstractive** summarization

For abstractive summarization, we will use a python implementation by "Huggingface Transformers". With this library we can easily implement the abstractive summarization tasks of BART and T5. It consists of a "pipeline" API which gives us easy access to the various pretrained models.

The ROUGE-1 scores for these are very close to the current state of the art implementation.

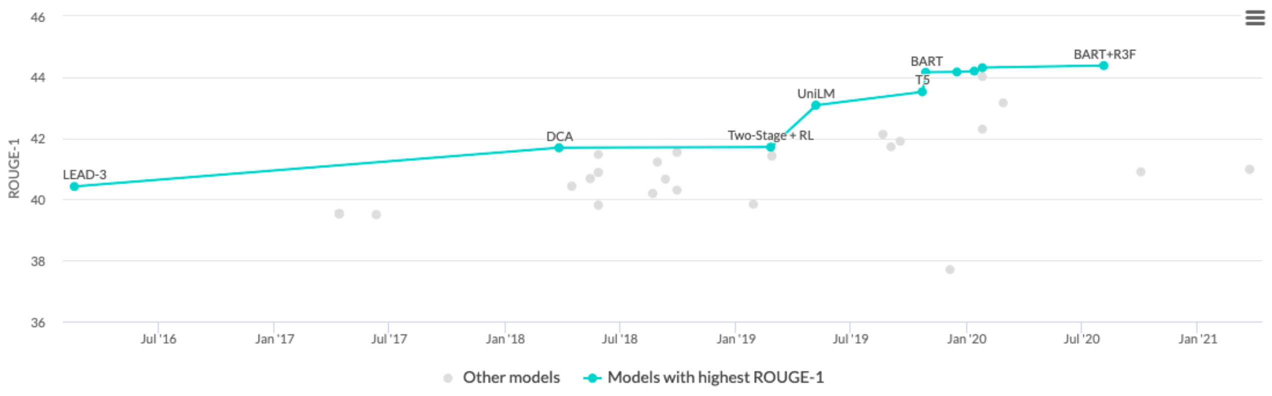


Figure 3 : CNN/Daily mail ROUGE-1 scores of different abstractive summary models.

ROUGE scores:
These are metrics used to evaluate the performance of a particular text summarization model. For this, a human generated summary is used as a reference with which the summary generated by the model is compared. The ROUGE-1 quantifies a unigram overlap of words between the reference and the model output. ROUGE-n quantifies an n-gram overlap. The models have to be compared against the same standard of dataset and the most popular one used is the CNN/Daily mail dataset. This dataset contains 300k articles and human generated summaries for each of these. In our dataset, we do not have a human generated reference.

Transformers:
The Transformer marks the major shift in language processing models. Initially these tasks used Recurrent Neural Networks which had some inherent disadvantages. They were slow to train, and there was the issue of vanishing gradients. Some of these issues were solved with the advent of LSTM networks where the gradients were remembered across longer sequences.
However, the LSTMs were slower than RNNs. Input data needed to be passed sequentially. The question needed to be solved was, how can we use the parallelization capabilities of modern GPUs to train sequential data. The transformer neural network architecture made this possible.
The transformer have an encoder decoder architecture like the RNNs. The word embeddings for all words in a sentence are determined simultaneously. The word embeddings are created by mapping similar meaning words to the same point in space. The word embeddings represent the word as a vector.

1. BART

a. Process

BART stands for Bidirectional Autoregressive Transformers. It is a denoising sequence to sequence architecture. It works by corrupting the text with some arbitrary noising function and then learning how to reconstruct the original text. This model can be treated as a general BERT model (Bidirectional Encoder Representations from Transformers) because it uses a standard sequence to sequence architecture with a bidirectional encoder, and also as a GPT2 model with a left to right decoder.

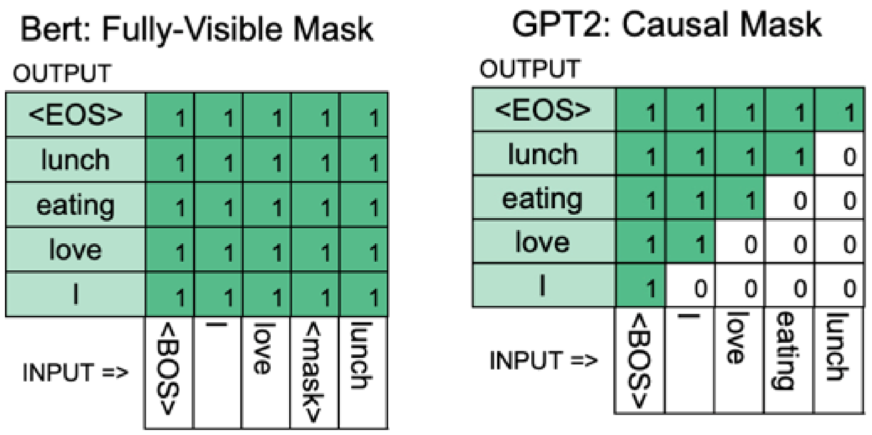


Figure 4 : BERT and GPT2 seq to seq architecture

Bart and T5 are both pretrained on tasks where spans of text are replaced by masked tokens. The model must learn to reconstruct the original document. The Figure 5 below from the BART paper explains it well:

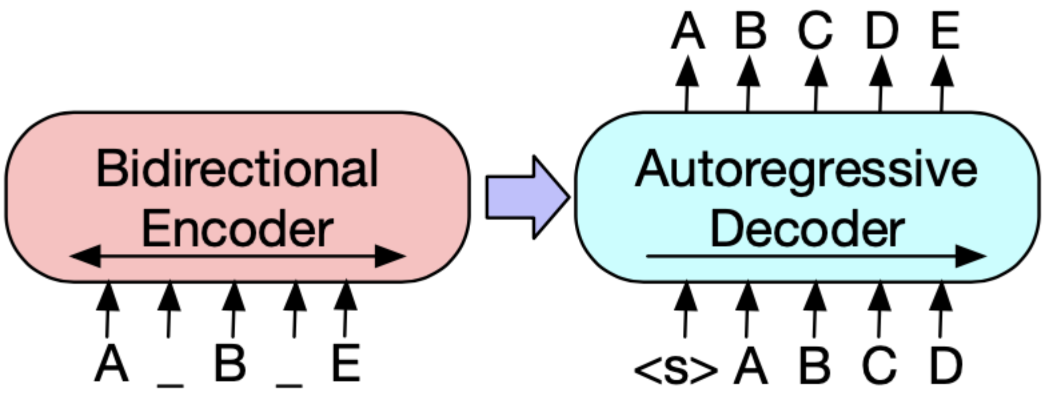


Figure 5 : Pre-trained model

In this example, the original document is A B C D E. the span [C, D] is masked before encoding and an extra mask is inserted before B, leaving the corrupted document 'A _ B _ E' as input to the encoder. The decoder (autogressive means "uses a causal mask") must reconstruct the original document, using the encoder's output and previous uncorrupted tokens.

In the huggingface transformers package, we need to import the “pipeline” module for the summarization task. By default, this refers to the BART summarization.

We can define a small function so that it can be applied across the dataframe.
The parameters to set are:

- *Input text:* individual rows from abstract column of the dataframe.

- *max_length*: maximum number of words in summary. Set 100 which summarizes in about 2-3 lines.
- *min_length*: minimum number of words in summary. Set to 5.
- *do_sample:* If we would like to sample different summaries generated for the same text. Set to 'False'.

The abstract summarization takes a long time to run. Running the model for 100 rows and taking the average time, we get a value of 11.5 secs for each abstract. That means to train the full dataset of over 80k abstracts, it would take approximately 10 days on google colab. However the standard runtime made available to us by colab is just for 12 hours of training. And within this time, resources may be reallocated and your session unexpectedly disconnected.

b. *Example*

Taking the same example abstract used in the previous steps, let us see the output generated by the BERT transformer model. The execution for this abstract of 6 sentences and 198 words took 10.1 secs:

"Precision agricultural technologies (PA) such as global positioning system tools have been commercially available since the early 1990s . But adoption studies show uneven adoption among farmers in the U.S. and Europe . This study aims to tackle a lingering puzzle regarding why some farmers adopt precision agriculture ."

c. *Observations:*

- The summaries generated by BART are well summarized and intuitive to read.
- It is not practical to implement on full dataset of 83799 abstracts.
- On average it takes 11.54 secs per abstract and is faster than T5
- The current state of the art implementation uses the BART architecture with some fine tuning. Specifically a model which is titled "Better Fine Tuning by Reducing Representational Collapse".
- The model was not capable of summarizing the biggest abstracts in the dataset. An error was generated with the following error: `Token indices sequence length is longer than the specified maximum sequence length for this model (3442 > 1024).` For the T-5 model this was `(3442 > 512).`

2. T5

a. *Process*

T5 for Text to Text Transfer Transformer is a transformer model trained on the Colossal Clean Crawled Corpus (C4). T5 model converts multiple NLP tasks into a text-to-text format. The tasks handled can be translation, question answering, classification and summarization. T5 is composed of 5 pre-trained weights :

- T5-small =60 million parameters
- T5-base =220 million parameters
- T5-large =770 million parameters
- T5–3B =3 billion parameters
- T5–11B = 11 billion parameters

So T5 is a unified multitask approach the task showed below can use the same model.

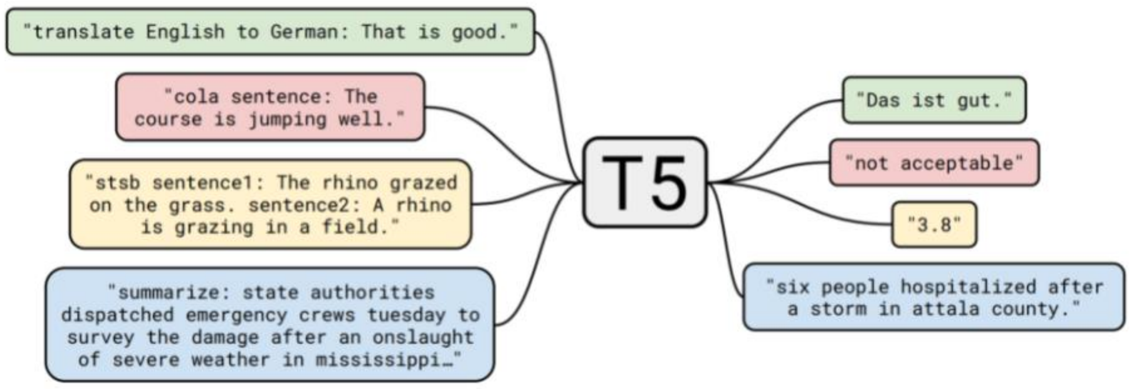

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Figure 6 : T5 model architecture

For the implementation of T5 we will use the huggingface transformers package. Using the “pipeline” model for summarization, we need to specify some more parameters to use T5-base model instead of the default BART model. The “model” and “tokenizer” parameters are set to “t5-base” and the “framework” is set to “pt” for pytorch. The other option available is “tf” which is tensorflow but we have implemented it with pytorch.

The defined function to pass to the summarizer have the same parameters as the previous BART implementation:

- *Input text*: individual rows from abstract column of the dataframe.
- *max_length*: maximum number of words in summary. Set 100 which summarizes in about 2-3 lines.
- *min_length:* minimum number of words in summary. Set to 5.
- *do_sample*: If we would like to sample different summaries generated for the same text. Set to ‘False’.

Even here the abstract summarization takes a long time to run. Running the model for 100 rows and taking the average time, we get a value of 15.2 secs for each abstract. That means to train the full dataset of over 80k abstracts, it would take approximately 14 days on google colab. For the sake of practicality, we have note executed on the entire dataset. However the code we have shared can be applied across the entire dataframe if you have the necessary computational resources.

b. *Example*

Taking the same example abstract (Abstract 1) used in the previous steps, let us see the output generated by the T5 transformer model. The execution for this abstract of 6 sentences and 198 words took 11.7 secs compared to BART which took 10.1 secs.

"adoption studies show uneven adoption among farmers in the U.S. and Europe . this study ai ms to tackle a lingering puzzle regarding why some farmers adopt precision agriculture as an approach to food production ."

c. *Observations:*

- The summaries generated by T5 not very intuitive to read.
- The sentences do not start with a capital letter.
- It is not practical to implement on full dataset of 83799 abstracts.
- On average it takes 15.24 secs per abstract compared to BART which takes 11.54 secs.
- Time for first abstract was 11.7 secs compared to BART which takes 10.1 secs.
- The model was not capable of summarizing the biggest abstracts in the dataset. An error was generated with the following: `Token indices sequence length is longer than the specified maximum sequence length for this model (3442 > 512)`

CONCLUSION

Through this project we were able to study the limitations of the different models and carry out comparisons between them. With the extractive models it was practical to obtain summaries for the entire dataset because they were not as time consuming as the abstractive methods. However these only highlighted the best sentences which represent the entire text.

The abstractive models produced summaries which generated new sentences to capture the meaning of the text. Here the BART model produced more intuitive summaries. The T-5 model summaries produced sentences which were not as well structured and did not begin with capital letters. Further, these abstractive models broke down with large text inputs and threw errors. T-5 had a greater sensitivity to larger texts and broke down if the token sequence length exceeded 512; compared to BARTs 1024.

The field is rapidly evolving and state of the art results are getting released every few months. Packages like the Huggingface transformers incorporate the abstractive models in easy to implement code and with access to a wide range of pretraining options.

The transformer models BERT,BART and T5 are best used with GPU enabled systems. Google colab is the best option to run training and inference. However, we faced difficulties in the free tier version, with several timeouts and runtime disconnections. There were also instances where we were not given access to the GPU for exceeding their resource allocations. For training these kind of models, access to a reliable and robust environment is essential to save time and effort. We must keep in mind however that these generate a heavy carbon footprint because of the long and resource intensive executions. A side by side comparison of inference on the first abstract is described below:

ext_textrank	The specific objective of this study is to examine the social and biophysical determinants of farmers' adoption of PA. This paper fills a research gap by including measurements of farmer identity—specifically their own conceptions of their role in the food system—as well as their perceptions of biophysical risks as these relate to the adoption of PA among a large sample of Midwestern U.S. farmers. The study has identified that farmer identity and perceptions of environmental risk do indeed influence PA adoption and that these considerations ought to be incorporated into further studies of PA adoption in other jurisdictions.
ext_bertsum	Precision agricultural technologies (PA) such as global positioning system tools have been commercially available since the early 1990s and they are widely thought to have environmental and economic benefit; however, adoption studies show uneven adoption among farmers in the U.S. and Europe. The findings also appear to highlight the social force of policy and industry efforts to frame PA as not only good for productivity and efficiency but also as an ecologically beneficial technology.
abs_bart	Precision agricultural technologies (PA) such as global positioning system tools have been commercially available since the early 1990s. But adoption studies show uneven adoption among farmers in the U.S. and Europe. This study aims to tackle a lingering puzzle regarding why some farmers adopt precision agriculture.
abs_sum_t5	adoption studies show uneven adoption among farmers in the U.S. and Europe. this study aims to tackle a lingering puzzle regarding why some farmers adopt precision agriculture as an approach to food production.

The time of execution for each of the models is mentioned below. As we can see the text rank algorithm is the fastest and the T5 model is the slowest. Also, within the first 100 abstracts, the T5 model encountered an error because of a large input length of the text.

Model	Iterative step	First 100 abstracts	First 10 abstracts	Average per abs
Extractive-TextRank	Similarity matrix	15.9s	1.87s	0.159s
Extractive-BERTSUM	Apply summarizer	9m 49s	59.1s	5.89s
Abstractive-BART	Apply summarizer	19m 46s	1m 59s	11.86s
Abstractive-T5	Apply summarizer	error	2m 30s	15s

REFERENCES

Text rank
https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/
Transformers
http://jalammar.github.io/illustrated-transformer/

CNN/Dailymail
https://www.tensorflow.org/datasets/catalog/cnn_dailymail

BERTSUM
Liu, Y. (2019). Fine-tune BERT for extractive summarization. arXiv preprint arXiv:1903.10318.
https://analyticsindiamag.com/hands-on-guide-to-extractive-text-summarization-with-bertsum/
https://medium.com/lsc-psd/a-bert-based-summarization-model-bertsum-88b1fc1b3177
http://web.stanford.edu/class/cs224n/reports/final_reports/report042.pdf
https://pypi.org/project/bert-extractive-summarizer/

T5
Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.
Zolotareva, E., Tashu, T. M., & Horváth, T. (2020). Abstractive Text Summarization using Transfer Learning.
https://medium.com/scavs-ai/summarize-news-article-in-2-minutes-using-t5-transformer-c63435083b3c
https://huggingface.co/transformers/model_doc/t5.html#overview
https://medium.com/scavs-ai/summing-up-dl-2-t5-text-to-text-transfer-transformer-eedf1c0bc63b
https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html
http://cs230.stanford.edu/projects_spring_2020/reports/38954132.pdf

BART
https://sshleifer.github.io/blog_v2/jupyter/2020/03/12/bart.html
http://cs230.stanford.edu/projects_spring_2020/reports/38866168.pdf