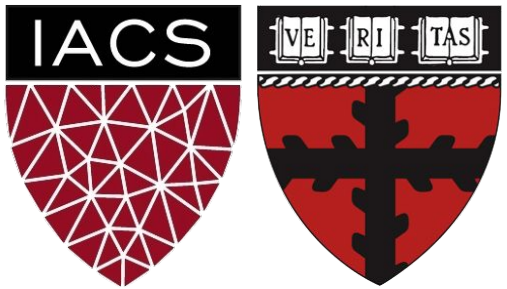
Project Garble

Milestone 2: Project Update Document

AC295

Joseph Kim, Malla Reddy Adaboina, Bharat Ramanathan



Outline

- Problem Definition
- Proposed Solution
- Project Scope
- Project Workflow
- Process Flow
- Data
- Models

Problem Definition

There are many lectures, podcasts, and other audio content online that we would like to consume. But in their current form, they're too long for us to go through them one by one.

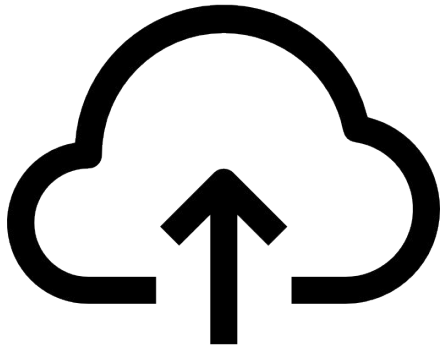
We would like to be able to quickly get the key takeaways from all these audio files efficiently.

Proposed Solution

We would like to build an app, accessible through a browser on mobile or desktop, that would allow any user to upload an audio file. The app would then generate a succinct text summary that capture the key points of the audio content.

Proposed Solution

1. Upload audio.



2. Receive text summary.

3. Read and profit.

Project Scope

Proof Of Concept (POC)

Two types of data: text and audio datasets.

- Long-form text. Should be “single voice” and include a summary.
- Long-form audio. Should be “single voice” and include a summary. Helpful to also have audio transcript.
- Conduct EDA across datasets
- Build baseline model
- Evaluate the performance of the baseline model

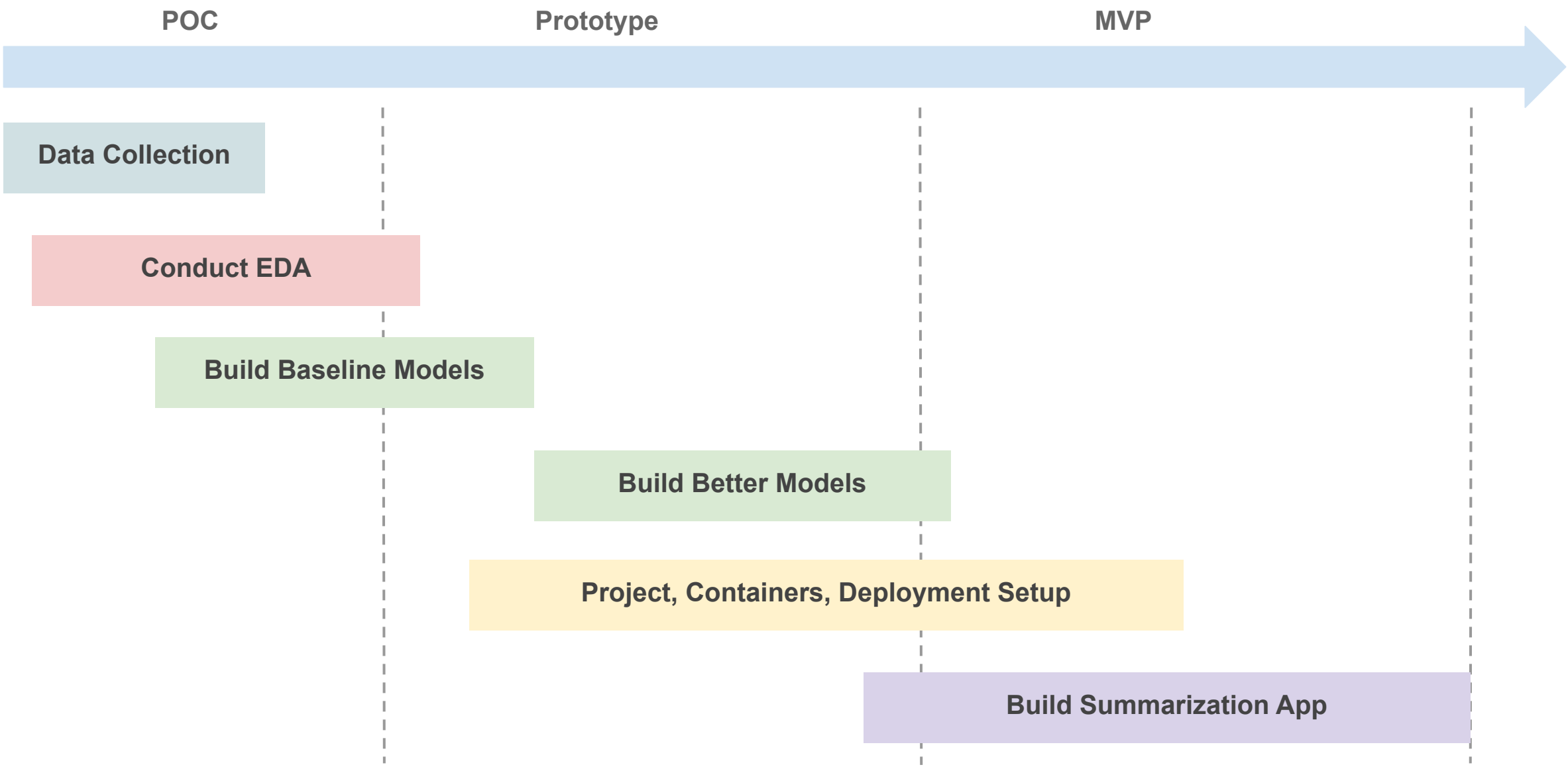
Prototype

- Create a mockup of screens to see how the app could look like
- Deploy one model to Fast API to service model predictions as an API

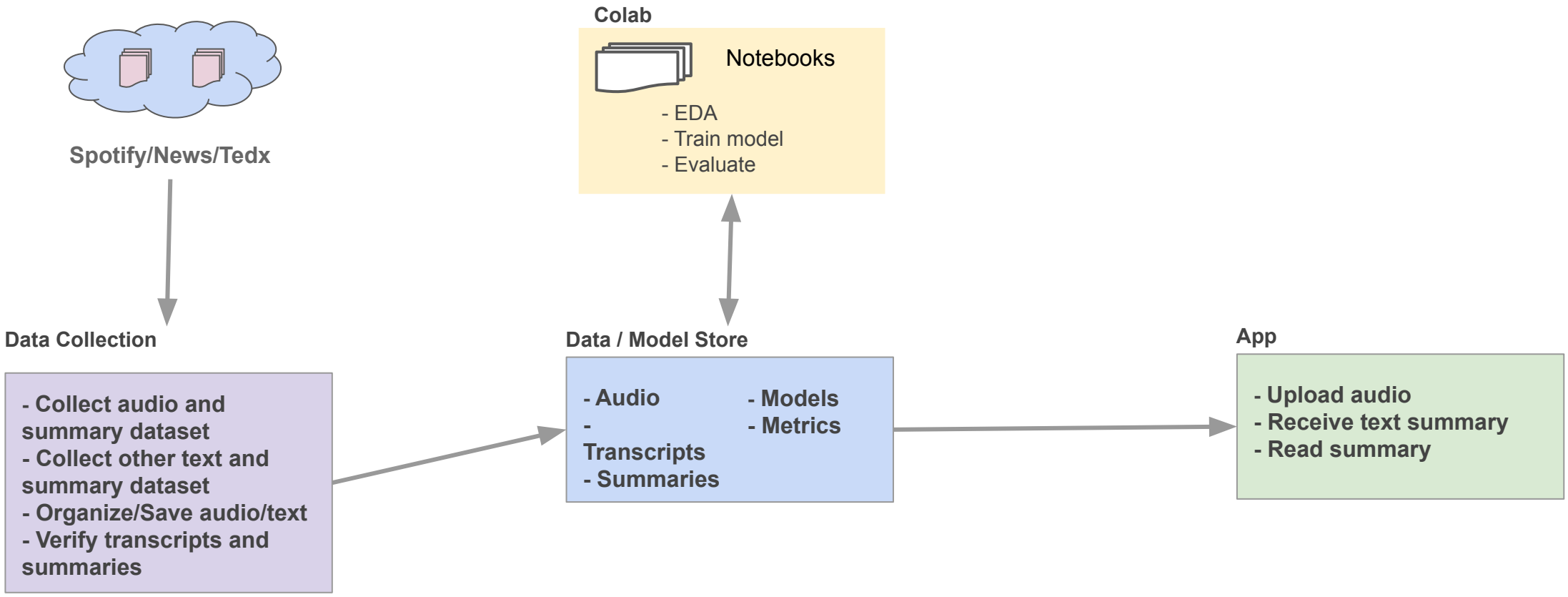
Minimum Viable Product (MVP)

- Create an integrated app (basic frontend) to generate text summaries for audio files
- Separate backend API server for converting audio to text and generating summaries

Project Workflow



Process Flow



Data: Sources

Name	Description	Source
AMI Meeting Corpus	100 hours of meeting recordings in English by mostly non-native speakers	https://github.com/gcunhase/AMICorpusXML
Common Crawl (CC) News	708241 English language news articles published between Jan 2017 and December 2019 from all over the world.	https://github.com/huggingface/datasets/tree/master/datasets/cc_news
CNN/Dailymail	300k unique news articles as written by journalists at CNN and the Daily Mail	https://github.com/abisee/cnn-dailymail
ICSI Meeting Corpus	70 hours of meeting recordings (multi-speaker meetings) in English. Includes non-native speakers.	http://www1.icsi.berkeley.edu/Speech/mr/
MediaSum	463.6K interview transcripts from NPR and CNN with summmaries	https://github.com/zcgzcgzcg1/MediaSum
Spotify	100k podcasts	https://podcastsdataset.byspotify.com/
TedX	All audio-video recordings of TED Talks uploaded to the official TED.com website until September 21st, 2017	TedX Kaggle Link
Extreme Summarization	226,711 news articles accompanied with a one-sentence summary	https://github.com/huggingface/datasets/tree/master/datasets/xsum

** for more details, check out the respective dataset_cards.md file for each dataset here: https://github.com/parambharat/AC215_projectgarble/tree/datasets/datasets/raw/supervised/summarization

Data: Quality

Name	Quality Rating	Additional Comments
AMI Meeting Corpus	4	Though this dataset mostly includes non-native speakers, it includes contains a wide range of other annotations including transcriptions, dialogue acts, topic segmentation, and extractive and abstractive summaries.
Common Crawl (CC) News	3	Large dataset useful for fine-tuning language models.
CNN/Dailymail	5	Large dataset useful for fine-tuning. Commonly used dataset for training models for the purposes of text summarization.
ICSI Meeting Corpus	1	Not ideal for a number of reasons: non-native speakers, varying audio quality, small data set, multiple speakers.
MediaSum	4	Large dataset of news articles with summaries of each. Perfect for training and testing. Written language though.
Spotify	5	This dataset covers a broad array of podcast types. It includes transcripts and summaries. As such, we can select the podcasts that specific fit our criteria.
TedX	4	TedX has the proper form that we're looking for: single speaker, long, and spoken language. Each Ted talk has a description which isn't necessarily the best summaries but should suffice.
Extreme Summarization	4	This news article dataset was explicitly compiled for the task of generating extractive and abstractive summaries. As such, this dataset has the "correct" summaries which can be used as the benchmark for testing. The downside is that the training text is "written" speech and not "spoken" speech.

Data Details (Doc Count & Mean)

ami	train	document	char_count	95.00	23670.63
			sentence_count	95.00	830.88
			sentence_density	95.00	0.13
			stopword_count	95.00	3557.26
			word_count	95.00	6177.65
		summary	char_count	95.00	7443.33
			sentence_count	95.00	142.48
			sentence_density	95.00	0.07
			stopword_count	95.00	1007.49
			word_count	95.00	1964.75

cc_news	train	document	char_count	566592.00	1991.58
			sentence_count	566592.00	22.48
			sentence_density	566592.00	0.06
			stopword_count	566592.00	170.73
			word_count	566592.00	396.85
		summary	char_count	566592.00	151.82
			sentence_count	566592.00	2.07
			sentence_density	566592.00	0.09
			stopword_count	566592.00	9.58
			word_count	566592.00	29.00

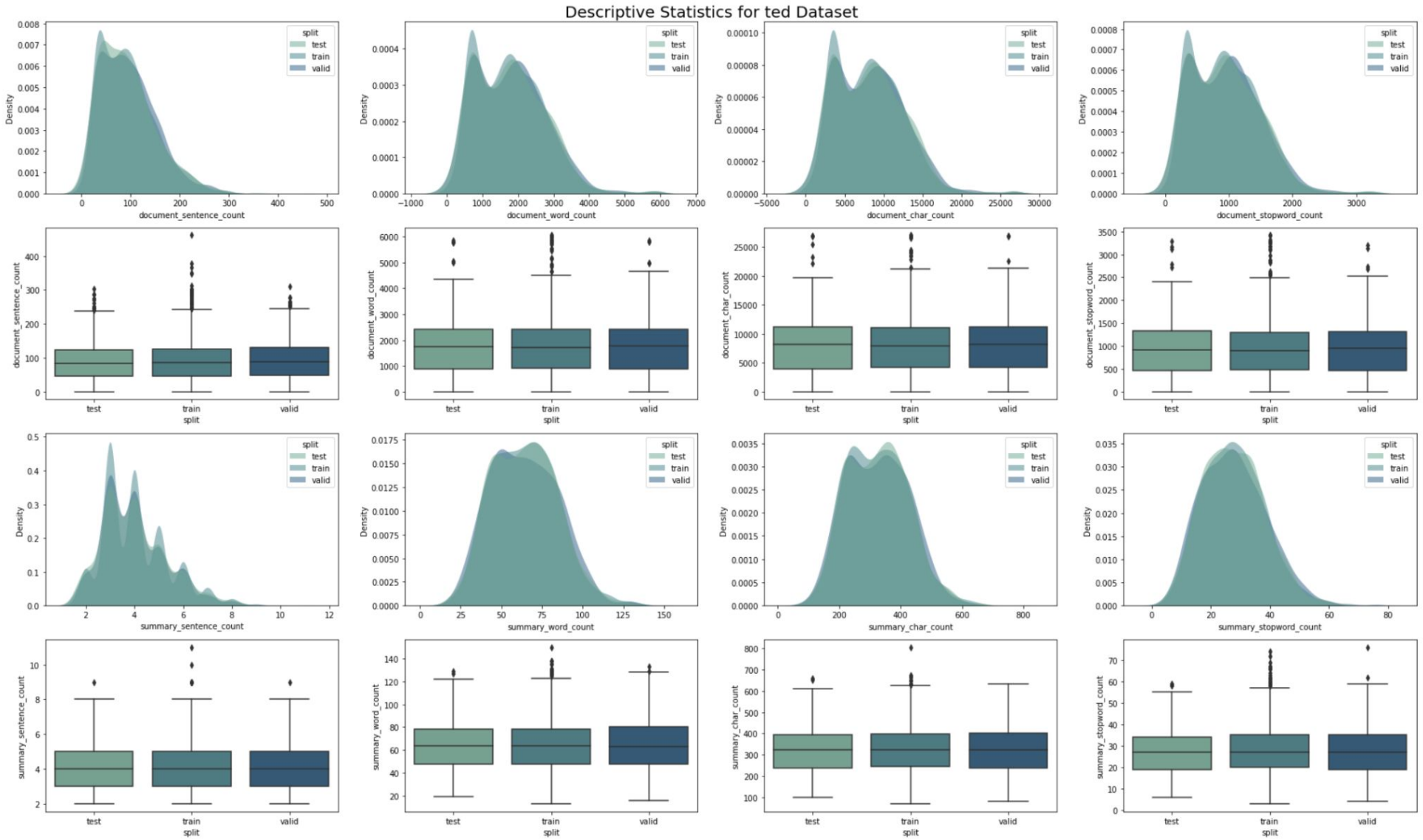
spotify	train	document	char_count	79173.00	24687.60
			sentence_count	79173.00	81.67
			sentence_density	79173.00	0.02
			stopword_count	79173.00	3255.05
			word_count	79173.00	5891.62
		summary	char_count	79173.00	409.57
			sentence_count	79173.00	3.50
			sentence_density	79173.00	0.05
			stopword_count	79173.00	32.76
			word_count	79173.00	76.06

cnn_dailymail	train	document	char_count	287113.00	3341.85
			sentence_count	287113.00	39.10
			sentence_density	287113.00	0.06
			stopword_count	287113.00	308.80
			word_count	287113.00	691.89
		summary	char_count	287113.00	244.16
			sentence_count	287113.00	3.83
			sentence_density	287113.00	0.08
			stopword_count	287113.00	19.02
			word_count	287113.00	51.57

ted	train	document	char_count	2828.00	8118.03
			sentence_count	2828.00	93.89
			sentence_density	2828.00	0.05
			stopword_count	2828.00	937.08
			word_count	2828.00	1759.56
		summary	char_count	2828.00	324.96
			sentence_count	2828.00	4.04
			sentence_density	2828.00	0.07
			stopword_count	2828.00	27.83
			word_count	2828.00	64.32

For more details: see https://github.com/parambharat/AC215_projectgarble/blob/datasets/notebooks/dataset_eda.ipynb

Sample Descriptive Visualization: TedX



Baseline Model

NLP Summarization Task

Use off-the-shelf SpaCy Textrank module to create extractive summaries

```
nlp = spacy.load("en_core_web_sm")
nlp.add_pipe("textrank")


def summarize_examples(examples):
    docs = nlp.pipe([" ".join(document) for document in examples["document"]])
    output_summaries = []
    for doc, summary in zip(docs, examples["summary"]):
        output_summary = doc._.textrank.summary(limit_sentences=len(summary))
        output_summaries.append([item.text for item in  output_summary])
    return {"predicted_summary": output_summaries}
```

```
# metric = datasets.load_metric("rouge")

def stringify_summaries(examples):
    return {"summary":["\n".join(document) for document in examples["summary"]],
    "predicted_summary":["\n".join(document) for document in examples["predicted_summary"]],
    }


data_splits = load_data_splits_from_dir(RAW_SUMMARIZATION_DATASETS_DIR)
features = datasets.features.Features({
    'document': datasets.Sequence(feature=datasets.Value(dtype='string', id=None), length=-1, id=None),
    'summary': datasets.Sequence(feature=datasets.Value(dtype='string', id=None), length=-1, id=None)
    })

baseline_scores = {}
for name, splits in tqdm(data_splits.items()):
    metric = datasets.load_metric("rouge")
    def compute_rouge(examples):
        result = metric.add_batch(predictions=examples["predicted_summary"], references=examples["summary"])
        return examples
    test_dataset = datasets.load_dataset("json", name, data_files={"test": splits["test"]}, features=features,)["test"]
    test_dataset = test_dataset.map(summarize_examples, batched=True)
    test_dataset = test_dataset.map(stringify_summaries, batched=True, remove_columns=test_dataset.column_names)


    test_dataset = test_dataset.map(compute_rouge, batched=True,)
    # test_dataset = test_dataset.to_pandas()
    baseline_scores[name] = metric.compute( use_stemmer=True)
```

Baseline Evaluation of Predicted Summaries

fmeasure

[-] **fmeasure { }**

('ami', 'rouge1', 'mid')	0.7684755597
('ted', 'rouge1', 'mid')	0.3119720149
('cc_news', 'rouge1', 'mid')	0.4249772021
('spotify', 'rouge1', 'mid')	0.2517385856
('cnn_dailymail', 'rouge1', 'mid')	0.48547556
('xsum', 'rouge1', 'mid')	0.2021605207
('icsi', 'rouge1', 'mid')	0.5895227987
('mediasumm', 'rouge1', 'mid')	0.2274605353

recall

[-] **recall { }**

('ami', 'rouge1', 'mid')	0.7684755597
('ted', 'rouge1', 'mid')	0.3119720149
('cc_news', 'rouge1', 'mid')	0.4249772021
('spotify', 'rouge1', 'mid')	0.2517385856
('cnn_dailymail', 'rouge1', 'mid')	0.48547556
('xsum', 'rouge1', 'mid')	0.2021605207
('icsi', 'rouge1', 'mid')	0.5895227987
('mediasumm', 'rouge1', 'mid')	0.2274605353

precision

[-] **precision { }**

('ami', 'rouge1', 'mid')	0.6464164899
('ted', 'rouge1', 'mid')	0.198752466
('cc_news', 'rouge1', 'mid')	0.2211000389
('spotify', 'rouge1', 'mid')	0.1722883916
('cnn_dailymail', 'rouge1', 'mid')	0.2509054662
('xsum', 'rouge1', 'mid')	0.1640921613
('icsi', 'rouge1', 'mid')	0.4477846356
('mediasumm', 'rouge1', 'mid')	0.1023154808

Next Steps

- Build a abstractive summary transformer model
- Evaluate model performance
- Tweak and iterate
- Build out API, frontend
- Integrate with Google speech-to-text
- Containerize apps and deploy
- End-to-end test