Project Proposal

Ananya Hari Narain

Automatic Generation of Podcast Summaries from Episode Transcriptions: Spotify Podcast Dataset Challenge

Project Originator: Dr Andrew Caines
Project Supervisors: Dr Andrew Caines and Dr Zheng Yuan
Director of Studies: Matthew Ireland
Overseers: Prof. Nic Lane and Prof. Richard Mortier

1 Introduction and Description of the Work

In recent years, podcasts have risen in popularity as a media source and Spotify is one of the current leaders in the podcast industry. As there are so many podcasts out there, it is often hard to find particular episodes that are tailored to one's interests and hence the need for effective automatic summarisation as a tool for analysis is growing.

Podcast summaries greatly inform which episodes a listener chooses to listen to, and human authored podcast descriptions do not always provide useful summaries, varying greatly in quality from podcast to podcast. My chosen project involves producing concise summaries based on episode transcriptions. The summaries should include the key contents of an episode, as per the rules of the Spotify Podcast Dataset Challenge. [1]

This is an interesting and challenging task because NLP summarisation has largely focussed on news articles containing formal language; podcasts tend to be dialogues with many colloquialisms and noisy with sponsorships and advertisements. Genre specific abstracts vary in style as well; sports podcast summaries are more factual and include names and in depth whereas true crime focussed podcasts tend to leave out information so as to provide an air of mystery and intrigue. I will need to ensure that the nature of the podcast summaries fit the genre of the episode.

Beyond the immediate focus, which is to produce a solution for the Spotify Podcast Dataset Summarisation Challenge, the work done here could be used as the basis for other dialogue summarisation. I plan on investigating extractive and abstractive summarisation methods and evaluating various models against each other.

[1] https://engineering.atspotify.com/2020/04/16/introducing-the-spotify-podcast-dataset-and-trec-challenge-2020/

2 Starting Point

- Python: I have experience coding in Python and have used it for various personal projects as well as in Part 1A in Scientific Computing and in Part 1B in Foundations of Data Science.
- PyTorch, TensorFlow: I have used PyTorch and TensorFlow before for personal projects and on an internship but want to re-familiarise myself with them to a greater extent.
- There is also some overlap with my project in the Natural Language Processing Part II unit of assessment which I am taking and I will be able to apply my knowledge to the dissertation. The scope of my dissertation, however, exceeds the course in the summarisation aspects and I will need to research a lot of domain specific aspects.
- I have read a number of relevant papers that pertain to summarisation, as well as previous entrants' entries to the Spotify Podcast Challenge. No code has yet been written for the project and I am starting from scratch.

3 Substance and Structure of the Project

Extractive models were the first types of models devised in order to tackle the problem of text summarisation; key sentences within the text are extracted and added to the summary. Hence, the generated summaries consist solely of sentences from the original text.

Abstractive models identify the important sections of the text, interpret and reproduce the content in a novel way. Sentences in the summary are generated rather than extracted or simply copied from the original text.

For the dissertation, I aim to train various extractive and abstractive models on the Spotify Podcast dataset and evaluate their relative performances.

1. Preprocessing the data:
    - The dataset is very large (13GB for the non-audio and 2GB for the audio components) so there will be a good deal of filtering that needs to be done. I plan on filtering out podcasts by their transcripts to create a shorter and consequently more manageable dataset for me to train my models on.
    - I will be removing any podcast transcripts that are too long, too short and contain very many adverts. In order to handle adverts I will need to train a simple classifier on an annotated set of advert-heavy transcripts. I plan on using an SVM for this that I can implement using the Python library sklearn.
2. BART model:
    - BART is a denoising autoencoder for pretraining sequence-to-sequence models. BART is trained by adding noise to text using a random noising function and learning a model to reconstruct the original text. [2]

[2] Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension by Lewis et al (2019)

- I appreciate that training a model from scratch may be very computationally expensive and seeing as I will be balancing work for my dissertation along with my units of assessment and other university-related work, I plan to finetune an existing BART model that is pretrained on a CNN/Daily Mail news summarization dataset [3] on the podcast transcript dataset.
- I plan to train my models on the MCS in around two days; training is expensive so I will be training for only 1 or 2 epochs in the interest of time. I plan to investigate the effectiveness of using a GPU or compute cluster for speeding this up; I plan on asking to use the HPC on the department's service level 2 account which would give me access to the GPU.
- In the case that I am unsuccessful with gaining access to the HPC, I can use Colab with the use of my Cambridge GSuite account to speed up the training of my BART model.

3. Pointer Generator model:

   - These models combine ideas from extractive and abstractive summarisation and are faster to train than purely abstractive models (in my case, the BART model). They also solve the shortcomings of abstractive models, that they often reproduce factual details incorrectly and include many repetitions.

4. Evaluation of the models

   - Metrics for evaluation:
     - Whilst the original podcast descriptions are not perfect, they serve as decent "ground truth" summaries as they are generated by the creator themselves. They may contain noise, such as the contact details of the creators as well as details of sponsorships but will contain enough of the key information to use to evaluate the auto-generated summaries.
     - I plan on using qualitative and quantitative metrics for evaluation as per the paper by Razapour et al (2020) [4]; On the quantitative side, I plan on using ROUGE-L to ascertain the longest common subsequence of words and ROUGE-N to ascertain the number of overlapping N-grams among other such scoring metrics.
     - I also plan on incorporating human evaluations in order to gain an understanding of the quality of the generated summaries compared with the episode itself. In order to do this, I will use my knowledge that I gained from Interaction Design in Part 1A and Further Human Computer Interaction in Part 1B. I will need to seek ethics approval from the department committee and have set aside a few weeks for this purpose in my timetable.

5. Extensions

   - Using a neural network for the preprocessor:
     Thus far I have only detailed a very basic classifier, but using a neural network for classification of advert-heavy podcast episodes has the potential to be both more accurate and increase performance as this will be introducing additional learned parameters instead of using kernels.

[3] https://huggingface.co/facebook/bart-large-cnn
[4] Rezapour, Rezvaneh, et al. "Spotify at TREC 2020: Genre-Aware Abstractive Podcast Summarization."

- Multimodal analysis:
  There is more information that could be extracted from the audio components of the datasets; for example, words and phrases that have more emphasis on them may need to be weighted more heavily when it comes to summarisation and such information could only be gleaned from an audio file. This is an area for exploration. I would be following some of the work explored by Baltrusaitis et al (2018) [5]
- Focussing on category aware summarisation:
  While genres cannot be obtained from the podcast dataset itself, I could try and extract them from RSS headers and form logical groupings of podcast genres and append them to the transcript in some way whilst fine-tuning/ training models. I may need to use a clustering algorithm, such as k-means to obtain a suitable set of genres.
- Reinforcement Learning model:
  Following the results of this paper by Paulus et al [6], I plan on training a neural network model that is trained in such a way that it combines standard supervised word prediction and reinforcement learning.
  This approach is another way to combat the shortcomings of purely abstractive models and will provide for interesting comparisons.

[5] Baltrušaitis, Tadas, Chaitanya Ahuja, and Louis-Philippe Morency. "Multimodal machine learning: A survey and taxonomy."

[6] https://arxiv.org/pdf/1705.04304.pdf

4 Success Criteria

The project will be considered a success if, given a transcript, my summarisation system produces a summary that is shorter than the original transcript and is in coherent English. I should have collected qualitative and quantitative evaluations that are indicative of a functional summarisation system and compared their performance on the datasets appropriately.
The success of my project is hence dependent on a few key components:

- Preprocessor:
  The preprocessor should be able to produce podcast transcripts that are devoid or light on advertisements. This can be evaluated by using standard metrics like precision and recall.
- BART model:
  The BART model should be implemented and should produce some sort of summary of a podcast given a transcript.
- Pointer Generator model:
  The Pointer Generator model should be implemented and should produce some sort of summary of a podcast given a transcript.
- Evaluating the models:
  The models should be compared against each other on certain metrics that I have specified earlier in the proposal, such as the ROUGE-n metrics.

5 Timetable and Milestones

I plan to divide my time into 2 week slots, in order to allow for sufficient granularity while not committing myself to a very tight schedule.

- Slot 0- 1st October - 18th October:
  - Read papers
  - Complete project proposal draft
  - Write proposal given feedback from the proposal
  - Set up GitHub repository, install libraries
  - Deadlines: Phase 1 proposal - 5th October, 12 noon
    Draft proposal - 8th October, 12 noon
    Proposal deadline - 15th October, 12 noon
  - Milestone: Proposal submission
- Slot 1- 19th October - 2nd November:
  - Familiarize myself with PyTorch, TensorFlow and any other libraries that I plan on using; I plan on working through some online tutorials
  - Implement the preprocessor; train a basic classifier and partition the dataset into training and test sets

  - Test the resulting dataset; write tests for specific cases
  - Make inquiries about the HPC and ask for permission to use it for the project
  - Milestone: Preprocessor written
- Slot 2- 3rd November- 17th November:
  - Research BART models and how they are implemented
  - Implement a BART baseline
  - Set up the project on MCS machine
  - Train model on MCS
  - If not completed, make inquiries about using the HPC.
  - Milestone: Baseline BART model written, confirm whether access to the HPC is possible or not.
- Slot 3- 18th November - 2nd December:
  - Finetune BART model; experiment with various filtering methods- random sentence filtering, truncation, TextRank, among others
  - Write scripts to evaluate the BART model
  - Milestone: BART evaluation scripts and experimentation done
- Slot 4- 3rd December- 17th December:
  - Implement Pointer Generator model
  - Train model on MCS
  - Write scripts to evaluate the model
  - Milestone: Pointer Generator evaluation scripts and experimentation done
- Slot 5- 18th December - 1st January:
  - Slack period; if not necessary, I will bring forward the evaluation of the models and look into the extension tasks ahead of time
  - Seek ethics approval from department committee
  - Milestone: All core methods implemented
- Slot 6- 2nd January - 16th January:
  - Evaluate the models based on the scripts written previously
  - If not done already, obtain the necessary approval from the department committee.
  - Milestone: Quantitative evaluation complete, approval from department committee gained.
- Slot 7- 17th January - 31th January:
  - Prepare progress report and presentation
  - Prepare for implementation of one of the extension tasks

  - Prepare questions for qualitative analysis and send to human evaluators.
  - Milestone: Deliver presentation and progress report

- Slot 8- 1st February - 15th February:
  - Rehearse for progress report presentation
  - Receive responses from the human evaluators, aggregate and document them.
  - Implement and evaluate the first extension

- Slot 9- 16th February - 2nd March:
  - Slack period; ensure that first extension is complete and outline plan for write-up before end of term
  - If I am ahead of time, I could consider implementing a second extension- potentially creating a UI could be finished quickly
  - Milestone: Complete first extension.

- Slot 10- 3rd March - 17th March:
  - Start writing up the dissertation; write the introduction and preparation chapters
  - Milestone: Draft for introduction and preparation chapters complete and sent to supervisors

- Slot 11- 18th March - 1st April:
  - Start writing the implementation chapter
  - Milestone: Draft for implementation chapter complete and sent to supervisors

- Slot 12- 2nd April - 16th April:
  - Write the evaluation chapter and ensure that I have acted on supervisors' feedback for any chapters written.
  - Milestone: Draft for evaluation chapter complete and sent to supervisors

- Slot 13- 17th April - 1st May:
  - Write the conclusions chapter and ensure that I have acted on supervisors' feedback for any chapters written
  - Milestone: Draft dissertation complete

- Slot 14- 2nd May - 13th May:
  - Slack period. Apply final touches to the dissertation and make sure that I have acted on all feedback that I have been given.
  - Milestone: Dissertation done!
  - Deadlines: Dissertation deadline - 13th May, 12 noon
    Source Code deadline - 13th May, 12 noon

6 Resources Declaration

I plan to use my own personal laptop:

- A Lenovo Legion computer with an Intel Core i7 CPU running at 2.6GHz, an Nvidia RTX 2070 GPU with 8GB of VRAM, 16GB of system RAM and 1TB of SSD storage. The machine runs Windows 10.

I accept full responsibility for any damage that happens to this machine and I have got sufficient money set aside to handle any machine repairs or replacements that are necessary. I can use the MCS machines in the meantime in the event of machine failure.
In addition, I plan on asking permission to use the HPC on the department's service level 2 account; if I am successful in gaining access to the HPC then this is also a resource I will be using.
I have taken a 3-pronged approach to protect myself against errors and data loss:

- I will be pushing changes made to my code and dissertation to a GitHub repository at the end of every day that changes are made. I will also be able to use this as a version control system.
- I will ensure that the contents of my git repository and dissertation are routinely backed up onto my University OneDrive; I will set up a scheduled task on my laptop.
- I will copy the contents of my git repository and dissertation onto a USB memory stick every two weeks, at the end of every time chunk scheduled in my timetable.

In order to carry out my dissertation I need access to the Spotify Podcast dataset and this has been obtained already. This was done by filling out a form on the official Challenge website [7]. These files are shared via the cloud and hence I will never lose access to them.

[7] https://podcastsdataset.byspotify.com/