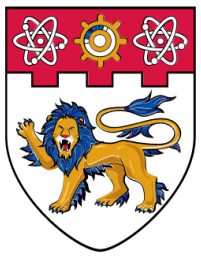

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Deep-Learning for Conversational Speech using Semantic Textual Analysis

Suthakar Shiny Gladdys

School of Computer Science and Engineering

Supervisor: Assoc. Prof. Chng Eng Siong

Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Computer Science of the Nanyang Technological University

CONTENTS

ABSTRACT

Automatic Speech Recognition (ASR) systems today have a prominent and widespread impact among software applications of different domains. They are usually embedded in the applications to provide user input to the main functionality, hence, acting as the cornerstone of these applications, especially potentially life-saving ones. However, most ASR systems today can only work effectively on formal speech input. They have a lot of room to fully understand speech of colloquial nature.

Focusing on English speech in the Singaporean context, this project aims to provide a solution for generating formal semantic equivalents of conversational sentences derived from speech. Thus, acoustic and language models of existing ASR systems can be trained with these mappings from conversational to formal text, thus acquiring better comprehension and performance when receiving informal speech input.

Furthermore, this project aims to analyse the semantic similarity performance of deep-learning models in terms of semantically similar formal sentence generation and their use of deep-learning techniques. The experimentation results show that the PEGASUS model performs better holistically. This report will present the proposed solution framework and lay out in detail the components of the project implementation.

ACKNOWLEDGEMENTS

This project was made possible by the support and guidance of the following people, who have helped me gain a very enriching and fruitful experience during this project.

I would like to acknowledge and express my deepest gratitude to Associate Professor Chng Eng Siong, my project supervisor, for his valuable guidance and advice throughout the duration of my project.

I would also like to express my sincere thanks to Mr Kyaw Zin Tun for his mentorship and assistance, and to all of the people whom I have worked with during this project.

Lastly, I would like to thank my family and friends for providing their continuous support throughout this project.

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Background

Automatic speech recognition (ASR) has become more prominent and reduced the need for human intervention in various areas of technology, especially artificial intelligence today. It has been the starting point of many essential applications such as virtual assistants, voice-enabled search engines and digital phone calls, with 27 percent of the world's online population making use of the voice search feature in their mobile phones in 2018 as reported by Google [1]. With human speech being the input for ASR, it has reduced the hassle for users by requiring them to spend lesser physical and mental effort when communicating with ASR-centred applications or devices. As a result, these ASR-centred applications and devices can be utilised by a wide range of target audience, such as people of relatively an older age, who are more prone to face physical and psychological fatigue and people who have suffer from visual impairment. Furthermore, they aid in maximising productivity for businesses, especially those who have limited manpower. Hence, ASR has indeed played a huge role in recent advancements in society.

1.2 Objectives and Significance

Current ASR systems are however only able to perform well on formal speech input as they are not able to fully comprehend speech of colloquial nature. This potentially deters people of certain demographics from using the ASR-centred tools due to their lower competency in formal language usage, as their intended meaning may not being understood completely or captured accurately. Previous studies [2] have shown that a lower education level did limit one's success in using ASR technology. This also applies to elderly people, who tend to face the problem of having lower focus and concentration in forming proper formal sentences.

One of the two main aims of this project is to generate formal sentences that possess similar semantic meanings to sentences obtained from conversational Singaporean English speech. The other main aim is to examine the semantic similarity performance of deep-learning models in the case of generating semantically similar formal sentences and study the deep-learning techniques involved.

The following are the motivations for the main aims of this project:

- Allow ASR systems to have a better comprehension of English speech impacted by sociolinguistic factors and speaking styles in the Singaporean context
- Enabling a wider demographic of the Singaporean population in using ASR-centred applications

The solution proposed will aid in training the acoustic and language models used by ASR systems such that they will be able to learn from

the mappings from conversational to formal sentences. Thus, there will be a better understanding of informal English speech in the Singaporean context by ASR systems, facilitating their high performance in the case of informal speech input.

1.3 Project Scope

The scope of this project is restricted to the English language in the Singaporean geographical context. The semantically similar formal sentence generation aspect of the solution should be able to accept multiple conversational sentence inputs at one time. This project will not provide details about how the ASR systems can be trained with the mappings from conversational sentences to formal ones.

2 LITERATURE REVIEW

2.1 Region-Influenced Speech

In past research [3], colloquial speech recognition was explored in the Algerian context. It investigated speech recognition performance in the case of Algerian regional-accented speech using the Arabic language. The speech evaluation dataset gathered consists of Arabic speech from both monolingual and bilingual speakers in various regions of Algeria. This was then used to train acoustic models in order to compare the accuracy among the regional accents. Including bilingual speakers' speech data was important as they tend to have more opportunities to use code-switching, which is essentially using varying languages at the same time to communicate. This is believed to occur because the speaker is not able to use a specific word or phrase in the current language they are using and hence, forced to deliver their intended message through another language.

The study has also looked into the issue of using the Algerian Modern Colloquial Arabic Speech Corpus (AMCASC), which served as the dataset for the study. It was realised that factors such as recording conditions mismatch and length mismatch impacted the accuracy of the acoustic techniques and perturbation sources such as code-switching should definitely be taken into account during speech evaluation.

2.2 Dialogue Generation

Using non-conversational text to enhance the diversity of dialogue utterances produced by sequence-to-sequence models has been looked into previously [4]. Due to the formal nature of non-conversational text, it usually encompasses a wider variety of domains and topics and is more abundant, compared to conversational text data. It also tends to have a higher level of variation of opinions and sentiments.

Following the training of the dialogue generation model on additionally non-conversational text, it was seen that dialogue responses produced had a greater level of variation in terms of both semantic and syntactic meaning. The concept of iterative back translation is used for the model in order to learn the mappings from the context to the unpaired non-conversational utterances.

2.3 Crawling-Based Corpora Construction

This study [5] discusses the features of corpora used for linguistic tasks analyses the advantages and disadvantages of web data. It has acknowledged the existence of widely known and used corpora such as the British National Corpus (BNC). Although the BNC might be of a very huge size, which is of a hundred million words, there still exists the possibility that some of the words in the English language appear rarely or do not even appear in the corpus. This sees the need for continual research and work in building corpora that aim for better inclusiveness

and appropriate representation of the elements of a language. The study has also highlighted that increasing the size of the corpus does still has a positive impact on the performance instead of the expected asymptotic results in cases of huge corpus size.

One advantage of utilising data from the web is that the web is the only source that can offer the high level of diversity required in corpora due to its high and wide usage. Moreover, web data contains content of genres that may not be available in written data sources. However, data from the web is prone to a large amount of noise, and one of the factors it could be attributed to is the way the data is being produced.

2.4 Semantic Similarity Measures for Short Texts

Past research [6] has been done on ways to implement the measurement of semantic similarity more meaningfully and efficiently. Text similarity is explored as a way to define semantic similarity. The study destails about multiple similarity methods for Natural Language Processing (NLP) tasks such as corpus-based methods and descriptive feature-based methods. The combination of these two types of methods are hybrid methods, which make use of both semantic and syntactic information about the text to find the text similarity. The study proposes a way to find text similarity between two short texts, which is using both semantic and string similarity. The string similarity plays a significantly bigger role in defining the text similarity and is measured by finding the longest common subsequence (LCS) between the two texts.

2.5 Speech Recognition Language Models

A slightly more recent study [7] on conversational vocabularies of both the Finnish and Estonian languages for speech recognition language models was looked into. Specifically, techniques to combat the inefficiency of language models brought about by huge vocabulary sizes were implemented. One observation was that class-based n-gram models performed better than standard word models in the case of large vocabularies. It was also found that recurrent neural network (RNN) language models based on statistical morphs have a better ability to utilize long contexts in the case of sub word models.

However, even whereby a large vocabulary size did not affect the performance of speech recognition systems, training time and memory consumption became setbacks when neural network language models were used. In addition, one of the obstacles faced during the conversational corpus development process was the abundance of web data present, which contributed a lot of noise due to the fact that a word can be written in alternate spellings. This was countered by implementing the interpolation of both class-based and word models.

3 PROPOSED APPROACH

3.1 Definitions and Assumptions

3.1.1 Definitions

Following are the definitions to the key terms used in this project:

- Conversational/Colloquial Text: Text obtained from an informal interaction setting
- Transcript: Text transcribed from audio/video
- Timestamps: Start and end times of a word/phrase/sentence
- Sentence Segmentation: Process of splitting a chunk of text into sentences
- Utterance: A sequence of words that does not necessarily have logical sentence structure
- Sentence: A sequence of words that have logical sentence structure

3.1.2 Assumptions

Following are the assumptions made in this project:

- Conversational text shall contain improper syntactic and logical structure, slang, fillers, non-specialist terminologies and interjections

3.2 Solution Framework

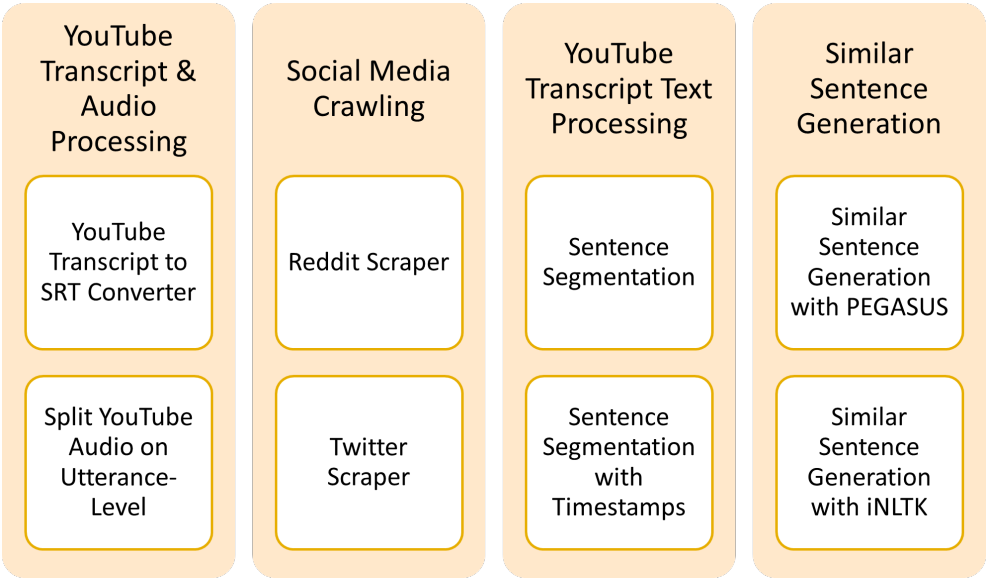

Figure 1: Solution Framework

As shown in Figure 1, the solution framework for this project involves four main stages. The first stage comprises of the file processing functionalities, followed by the next stage, which consists of crawling processes for social media sites Reddit and Twitter. The third stage handles the sentence segmentation process for the conversational text obtained from YouTube. The final stage is the implementation of generating semantically similar formal sentences using the PEGASUS and iNLTK models.

3.3 Project Methodology

3.3.1 Conversational Text Acquisition

The first stage of the project methodology consists of obtaining text from conversational English speech in the Singaporean context. The text was crawled from YouTube videos, mainly from the targeted medical and health domains. This was because the crawled data would be relevant to and can be used for ASR systems employed in emergency response applications.

```
0.60 4.48       [Music]
4.48 5.12       today
5.12 7.52       a million wildlife species are facing
7.52 10.24      the threat of extinction
10.24 13.04     what's frightening they're dying off
13.04 14.08     faster than
14.08 17.52     ever before
17.52 19.52     it's just a really unnerving feeling
19.52 21.68     knowing that somewhere down the road
21.68 23.84     the animals that we have today may only
23.84 25.44     exist behind glass like what we're
25.44 27.60     seeing here
27.60 30.32     but we live in a tiny city-state i
30.32 30.96     wonder
30.96 33.20     how can we possibly make biodiversity an
33.20 35.20     issue that matters to more people in
35.20 37.20     singapore
37.20 39.84     i'm shushan born and bred in singapore
39.84 41.44     and in my time as a journalist
41.44 43.52     my favorite stories to cover were always
43.52 45.36     the ones about nature
45.36 47.12     i discovered a secret sanctuary for
47.12 48.72     gibbons rescued from poachers in
48.72 51.28     malaysia
51.28 53.76     explored the mountains of borneo we
```

Figure 2: YouTube Transcript with Utterance-Level Timestamps

The text obtained was in the form of transcripts of the YouTube videos with timestamps on the utterance level using YoutubeTranscriptAPI [8].

Figure 2 shows an example of a YouTube transcript with utterance-level timestamps.

```
WEBVTT
Kind: captions
Language: en

00:00:00.060 --> 00:00:31.429 align:start position:0%

[Music]

00:00:31.429 --> 00:00:31.439 align:start position:0%


00:00:31.439 --> 00:00:33.910 align:start position:0%

it's<00:00:31.599><c> when</c><00:00:31.760><c> we</c><00:00:31.920><c> start</c><00:00:32.160><c>

00:00:33.910 --> 00:00:33.920 align:start position:0%
it's when we start off to design a vc


00:00:33.920 --> 00:00:36.950 align:start position:0%
it's when we start off to design a vc
first<00:00:34.320><c> um</c><00:00:35.040><c> it</c><00:00:35.200><c> has</c><00:00:35.360><c> to<

00:00:36.950 --> 00:00:36.960 align:start position:0%
first um it has to be quickly built and


00:00:36.960 --> 00:00:38.709 align:start position:0%
first um it has to be quickly built and
yet<00:00:37.280><c> it's</c><00:00:37.360><c> got</c><00:00:37.520><c> to</c><00:00:37.600><c> be<
```

Figure 3: YouTube Transcript with Word-Level Timestamps

Obtaining transcripts of YouTube videos which contained the word-level timestamps was also explored, which was using YouTube-dl [9]. Figure 3 shows an example of a YouTube transcript with word-level timestamps. Since the transcripts have repeating lines and inconsistent formatting as seen in Figure 3, they were processed using an existing script to remove the redundancies and be in a format that can be easily readable and manipulated.

The following are the additional tasks implemented for other uses related to acquiring conversational text for ASR systems:

- YouTube Transcript to SubRip Subtitle (SRT) File Conversion

- Splitting YouTube Audio on Utterance-Level
- Reddit and Twitter social media scraping

The generation of SRT files from YouTube video transcripts was implemented such that the SRT files can be used in subtitle editors, for example, Subtitle Edit [10], to test subtitle synchronisation with the YouTube audio. Additionally, YouTube audio splitting on the utterance level was executed, which can be used in ASR systems to examine the accuracy of speech recognition on utterances.

title	numComm	upVotes	author	subreddit	isSponsore	link	comme	mall	area
Bizen Okayama Wagyu Steakhouse @ Plaza Singapura	1	6	midasp	r/Singapor	FALSE	http	{}	Plaza Singapura	central
Let's discuss - Perhaps people who are not ill can use these reu	35	39	_LadyGala	r/singapor	FALSE	http	{}	Plaza Singapura	central
Contrary to popular belief, the Plaza Singapura has a public res	0	1	farklinkbot	r/fark	FALSE	http	{}	Plaza Singapura	central
Five Guys to open first Singapore outlet at Plaza Singapura	56	158	vaultofech	r/singapor	FALSE	http	{}	Plaza Singapura	central
Plaza Singapura looking chill during cb	1	34	djmpence	r/singapor	FALSE	http	{}	Plaza Singapura	central

Figure 4: Scraped Reddit Posts Containing Names of Malls in Singapore

id	date	tweet	road
1.32198E+18	2020-10-30 01:09:	And half of ur bukit timah is clementi.	bukit timah
1.32196E+18	2020-10-29 23:38:	@eaglesbirdies @StephenMcDonell You can only get an exit visa on certain	bukit timah
1.32185E+18	2020-10-29 16:15:	@LakiBean Hahah i knew about it quite long ago and even how his fave	bukit timah
1.32183E+18	2020-10-29 14:46:	@NazmiHanafiah Hang singapore kat mana ? Bedok ? Bugis junction ? Bukit	bukit timah
1.32138E+18	2020-10-28 09:07:	BAIKLAH BACK TO BUKIT TIMAH YAY WEEHOOOOO	bukit timah

Figure 5: Scraped Tweets Containing Names of Roads in Singapore

For the purpose of enriching language models with conversational sentences revolving around food and locations in Singapore, text from the social media sites Twitter and Reddit were also scraped. The Twitter posts, also known as tweets, and Reddit posts contained the targeted keywords, which were names of food, roads and shopping malls in Singapore as seen in Figures 4 and 5.

3.3.2 Text Pre-Processing and Normalisation

With the text obtained from the conversational speech from YouTube, some pre-processing and normalization steps were applied to make the text standardized for further processing and manipulation. The first and foremost step of the processing is that all the text is converted to lowercase before being processed for other attributes.

Order	Attribute(s) Processed	Example Input and Output
1	Salutations, Symbols, Event Fillers	'mr' → 'mister' '&' → 'and' '[music]' → ''
2	Numbers in Numerical Format	'7' → 'seven' '18.2' → 'eighteen point two'
3	Accented Characters	'é' → 'e' 'ü' → 'u'

Table 1: Attributes Processed in Conversational Text

The above table shows which attributes of the text that are targeted and processed, and also in which order that they are processed.

Attribute Not Processed	Example
Hyphenated Words	'empty-handed'
Contractions	'could've'

Table 2: Attributes Not Processed in Conversational Text

However, some attributes are not processed to prevent loss in meaning of the text and to preserve the original speech form of the text.

3.3.3 Sentence Segmentation

The presence of punctuation in text is important in natural language processing tasks as it aids in signaling the boundaries of sentences in the text and makes the text more logical and readable.

Looking at the nature of the conversational text obtained from YouTube, it does not contain punctuation. Moreover, due to the continuous nature of speech in videos, it was found to be hard to identify where a sentence starts and ends in the text. Hence, it is not possible to define specifically the original intended sentence boundaries in the text. In addition, the option of considering the utterances in transcripts with utterance-level timestamps as sentences was also explored. However, the way the text was divided into the utterances was dependent on which group of words appeared at once as the video subtitles, and each utterance was of about the same length. Hence, the utterances could not be considered as proper sentences as they did not have the structural properties of a sentence and were not logical as sentences.

```
0.60 4.48       [Music]
4.48 5.12       today
5.12 7.52       a million wildlife species are facing
7.52 10.24      the threat of extinction
10.24 13.04     what's frightening they're dying off
13.04 14.08     faster than
14.08 17.52     ever before
17.52 19.52     it's just a really unnerving feeling
19.52 21.68     knowing that somewhere down the road
21.68 23.84     the animals that we have today may only
23.84 25.44     exist behind glass like what we're
25.44 27.60     seeing here
27.60 30.32     but we live in a tiny city-state i
30.32 30.96     wonder
30.96 33.20     how can we possibly make biodiversity an
33.20 35.20     issue that matters to more people in
35.20 37.20     singapore
37.20 39.84     i'm shushan born and bred in singapore
39.84 41.44     and in my time as a journalist
41.44 43.52     my favorite stories to cover were always
43.52 45.36     the ones about nature
45.36 47.12     i discovered a secret sanctuary for
47.12 48.72     gibbons rescued from poachers in
48.72 51.28     malaysia
51.28 53.76     explored the mountains of borneo we
```

Figure 6: YouTube Transcript before Sentence Segmentation

```
today a million wildlife species are facing the threat of extinction
what's frightening they're dying off faster than ever before
it's just a really unnerving feeling knowing that somewhere down the road
the animals that we have today may only exist behind glass
like what we're seeing here but we live in a tiny city-state i wonder
how can we possibly make biodiversity an issue that matters to more people in singapore
i'm shushan born and bred in singapore and in my time as a journalist my favorite stories to co
i discovered a secret sanctuary for gibbons rescued from poachers in malaysia explored the moun
we reached ten of our journey and spend my holidays in other natural wonders abroad
covert nineteen put an end to my trips abroad but being stuck in singapore has turned out to be
this is looking at us with his eyes wide open
oh it's so cute in this two-part documentary i come face to face with some of the rarest specie
forests blows my mind really and meet the tenacious men and women in the uphill battle to save
these crabs are only found in singapore in the last two hundred years
we have lost nearly a third of our native wildlife
can we make it in time to stop the remaining species from disappearing forever
it was thought to be the last one of its tribe in the history of humanity
we've been exploiting the bounties of nature for our needs
you could almost say it's in our nature but have we come to a point where we've gone too far
what do we stand to lose
when our own flora and fauna go extinct and what role can our tiny city-state possibly play in
that's what i'll try to find out in it's in our nature
i love spending my mornings here but today i'm not here for a casual hike
i'm taking part in a competition a five-hour bird race
```

Figure 7: YouTube Transcript after Sentence Segmentation

DeepSegment [11], a Python library, was found to counter this issue. It allowed for unpunctuated text to be transformed into proper sentences that have the logical structure of a sentence and are of appropriate length. Hence, sentence segmentation of the text by DeepSegment was carried out using a Python script. Figures 6 and 7 show a transcript before and after the process of sentence segmentation.

```
<00:00:04.470> <00:00:10.559>    today a million wildlife species are facing the threat of extinction
<00:00:10.559> <00:00:17.760>   what's frightening they're dying off faster than ever before
<00:00:17.760> <00:00:21.840>   it's just a really unnerving feeling knowing that somewhere down the road
<00:00:21.840> <00:00:25.119>   the animals that we have today may only exist behind glass
<00:00:25.119> <00:00:30.310>   like what we're seeing here but we live in a tiny citystate
<00:00:30.310> <00:00:37.440>   i wonder how can we possibly make biodiversity an issue that matters to more peopl
<00:00:37.440> <00:00:45.520>   i'm shushan born and bred in singapore and in my time as a journalist my favorite
<00:00:45.520> <00:00:53.750>   i discovered a secret sanctuary for gibbons rescued from poachers in malaysia expl
<00:00:53.750> <00:01:02.480>   we reached ten of our journey and spend my holidays in other natural wonders abroa
<00:01:02.480> <00:01:07.520>   covert nineteen put an end to my trips abroad but being stuck in
<00:01:07.520> <00:01:10.720>   singapore has turned out to be an eyeopening experience
<00:01:10.720> <00:01:13.920>   this is looking at us with his eyes wide open
<00:01:13.920> <00:01:17.520>   oh it's so cute in this twopart documentary
<00:01:17.520> <00:01:25.990>   i come face to face with some of the rarest species in the world right here in sir
<00:01:25.990> <00:01:33.680>   forests blows my mind really and meet the tenacious men and women in the uphill ba
<00:01:33.680> <00:01:37.840>   these crabs are only found in singapore in the last two hundred years
<00:01:37.840> <00:01:41.280>   we have lost nearly a third of our native wildlife
<00:01:41.280> <00:01:45.680>   can we make it in time to stop the remaining species from disappearing forever
<00:01:45.680> <00:01:52.069>   it was thought to be the last one of its tribe
<00:01:52.069> <00:01:53.920>    in the history of humanity
<00:01:53.920> <00:01:57.680>   we've been exploiting the bounties of nature for our needs
<00:01:57.680> <00:02:04.880>   you could almost say it's in our nature but have we come to a point where we've go
<00:02:04.880> <00:02:16.879>   what do we stand to lose when our own flora and fauna go extinct and what role car
<00:02:16.879> <00:02:39.920>   that's what i'll try to find out in it's in our nature
<00:02:39.920> <00:02:46.080>   i love spending my mornings here but today i'm not here for a casual hike
<00:02:46.080> <00:02:51.120>   i'm taking part in a competition a fivehour bird race
```

Figure 8: YouTube Transcript after Sentence Segmentation with Timestamps

The sentence segmentation implementation was also adapted to include timestamps of the newly-formed sentences. The need for acquiring timestamps of the newly-formed sentences is so that it can allow for further tasks such as generating SRT files and splitting audio files on the sentence level to be done. Hence, an additional Python script was used to carry out the same sentence segmentation process but it additionally worked out timestamps of the newly-formed sentences from transcripts with word-level timestamps. This explains the need to crawl transcripts with word-level timestamps in addition to the transcripts with utterance-level timestamps in the first stage of the project methodology.

Figure 8 shows the transcript in Figure 6 after the process of sentence segmentation with timestamps.

3.3.4 *Deep-Learning Models for Semantically Similar Formal Sentence Generation*

With the aim of generating formal sentences that are similar in meaning to the conversational sentences, the two following deep-learning models were experimented with:

- PEGASUS [12]
- iNLTK [13]

The PEGASUS model works mainly on the concept of abstractive summarization. Abstractive summarization has the main aim of processing a paragraph into fewer sentences, summarizing the main idea and information of the original chunk of text. Since the summarization is abstractive, the generated sentences are not extracted from parts of the original text but rather constructed by the model itself. PEGASUS can be used for the purpose of sentence generation as well. The PEGASUS model, finetuned for the task of paraphrasing, is used in this case.

The iNLTK model is based on the ULMFiT [14] language model and performs a variety of tasks, such as language identification, sentence encoding, similar sentence generation. It is designed to work on various Indian languages as well.

3.4 Tools and Technologies

3.4.1 Environment Setup

Hardware Requirements	- 16.0 GB RAM - Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
Operating System	Windows 10
Integrated Development Environment (IDE)	Visual Studio Code using Microsoft Python extension
Conda Package and Environment Management System	Conda version 4.10.3
Web Browser	Google Chrome

Table 3: Environment Setup

The above table details the environment setup used for this project.

3.4.2 YouTube

YouTube is one of the world's largest video-sharing platforms. It also serves as a social media site, where users can upload mainly videos and other types of content. Using speech-to-text Application Programming Interfaces (APIs) on YouTube videos, we can make use of the large amount of text data available from a wide variety of languages and domains.

3.4.3 Subtitle Edit

Subtitle Edit is an open-source software which provides the functionality of a subtitle editor. It supports multiple file formats such as SRT, MicroDVD and SAMI. FFmpeg is additionally required if one wants to generate scene

changes in Subtitle Edit. There are numerous uses of Subtitle Edit, of which the main ones are:

- Add and edit subtitles of videos
- Adjust subtitles for synchronisation with videos using features such as changing frame rate
- Modify the type of appearance of subtitles in videos
- View audio waveform and video together with subtitles

3.4.4 Pydub

Pydub [15] is a Python library that allows users to handle and apply various functions on audio files, such as MP4, OGG and WAV files. It works with Ffmpeg [16] to to carry out some of its functions.

3.4.5 YoutubeTranscriptAPI

YoutubeTranscriptAPI is an API that allows you retrieve subtitles of YouTube videos in the form of transcripts with utterance-level timestamps, whether it is the automatically generated subtitles or the original subtitles. It also works on subtitles available in other languages.

3.4.6 Youtube-dl

Youtube-dl is a Python library that one can use to also obtain transcripts that contain subtitles of videos from YouTube. However, it is different from YoutubeTranscriptAPI in the sense that it is able to generate the

transcripts containing word-level timestamps. Furthermore, it can be used in the command line. It works on videos from video websites other than YouTube as well.

3.4.7 DeepSegment

DeepSegment is a Python library that acts as a sentence segmenter. It has the ability to carry out sentence segmentation on both punctuated and unpunctuated chunks of text. Users shall be able to finetune the DeepSegment model with their data and parameters such as learning rate and batch size can be manipulated in the process.

3.4.8 PyTorch

PyTorch [17] is a Python library, providing implementations for neural networks. In this project, the library is used for tensor computation for the deep-learning models.

4 IMPLEMENTATION

4.1 YouTube Transcript to SRT Converter

The format of a YouTube transcript with utterance-level timestamps is such that each line will contain the start and end time of an utterance, followed by the utterance itself. Such a transcript is then transformed into a file that is of SRT format, whereby each original utterance and its timestamps are considered a subtitle object in the SRT file. Each subtitle object consists of the following attributes:

- Subtitle object number
- Start and end times of subtitle object separated by '–>'
- Text of subtitle object spanning one or more lines
- Blank line signalling end of subtitle object

```
1
00:00:00,670 --> 00:00:08,320
[Music]

2
00:00:08,320 --> 00:00:10,720
last year this time

3
00:00:10,720 --> 00:00:14,160
i was traveling the world investigating

4
00:00:14,160 --> 00:00:17,040
food mysteries disappearing fish in

5
00:00:17,040 --> 00:00:20,560
indonesia
```

Figure 9: SRT File

An SRT file also has a different timestamp format from that of the YouTube transcript. Hence, the start and end times of the utterances were required to be modified from the original '[Seconds].[Milliseconds]' format to the 'Hours:Minutes:Seconds,Milliseconds' format. Figure 9 shows an example of a SRT file.

This implementation as a Python script is able to work on a folder containing multiple YouTube transcripts and produces the corresponding SRT files in a new folder under the same directory.

4.2 Split YouTube Audio on Utterance-Level

For the task of splitting YouTube audio files in terms of utterances, we require both the YouTube audio file and the corresponding transcript with utterance-level timestamps.

```
audio = AudioSegment.from_wav(os.path.join(wav_folder, name + '.wav'))

for index, row in df.iterrows():
    newAudio = audio[row.st_final:row.et_final]
```

Figure 10: Code Snippet for Segmentation of Audio

The Python script for this implementation refers to the the transcript for the start and end times of each utterance, and using that, it segments the corresponding audio file with the help of the AudioSegment module from the Pydub library as seen in Figure 10. Hence, a separate audio file is produced for each utterance in the .wav file format.

The Python script can work on multiple YouTube audio files and their corresponding transcripts. It will output the segmented audio files in a new folder under the same directory, with the each file name indicating the original audio file name and the start and end time of the utterance in terms of milliseconds.

4.3 Reddit Scraper

Reddit is one of the widely-known social media sites today and it can be scraped to get Reddit posts for text data collection and research purposes. Most Reddit scrapers make use of Python Reddit API Wrapper (PRAW) [18], which requires valid Reddit account credentials. Hence, the process of scraping Reddit without the need for a Reddit account was implemented with a Python script. This reduces the hassle of creating a Reddit account for users who want to scrape Reddit for relevant data but do not have an existing Reddit account.

The implemented Reddit scraper takes in the following inputs:

- Search string (String to be searched in Reddit)
- Number of Reddit posts to be scraped
- Flag for option to scrape comments of Reddit posts

```
self.search_url = "https://gateway.reddit.com/desktopapi/v1/search?q={}&after={}"
self.comments_url = "https://gateway.reddit.com/desktopapi/v1/postcomments/{}"
```

Figure 11: Code Snippet for API for Reddit Scraping

The scraping methodology uses public API endpoints as seen in Figure 11. The script outputs a JSON file, in which each JSON object represents a Reddit post and its information.

4.4 Twitter Scraper

Twitter is another popular social media site that poses as a platform for hosting a huge amount of text data, mainly in Twitter posts, which are also known as tweets. It has its own API that allows users to retrieve Twitter data. However, it has quite a number of limitations, notably limiting the available tweets to the ones posted in the last seven days and the limiting the maximum tweets scraped per unit time. Moreover, a Twitter developer account is required in order to use the API. Thus, a Twitter scraper was implemented in the form of a Python script to override these limitations. As a result, it does not have any restrictions on the time period of the tweets that can be scraped and the scraping speed, and it does not require an existing Twitter developer account.

The implemented Twitter scraper takes in the following inputs:

- Search string (String to be searched in Twitter)
- Number of tweets to be scraped

4.5 Sentence Segmentation

The task of sentence segmentation includes the execution of the text normalisation as well. The sentence segmentation process occurs after the text is normalised.

The sentence segmentation works on YouTube transcripts with utterance-level timestamps. For each YouTube transcript, all the utterances are concatenated into a singular chunk of text. The sentence segmentation is performed on the chunk of text with the help of the DeepSegment library, and a list of sentences are returned.

```
segmenter = DeepSegment('en')
sent_list = segmenter.segment_long(para)
```

Figure 12: Code Snippet for Sentence Segmentation

The 'segment_long' method is used as it is suitable for relatively longer chunks of text.

```
# split words together with their end times
df['times'] = df['times'].str.split(' ')
s = df.apply(lambda x: pd.Series(x['times']), axis=1).stack().reset_index(level=1, drop=True)
s.name = 'times'
df = df.drop('times', axis=1).join(s)
df = df[df['times'].str.contains("<") == True]
df = df.reset_index(drop=True)
df['times_2'] = df['times'].str.split('<')
df.loc[:, 'word'] = df.times_2.map(lambda x: x[0])
df.loc[:, 'end_time'] = '<' + df.times_2.map(lambda x: x[1])

# get start time of each word
df['start_time'] = df['end_time'].shift(1)
df['start_time'].iloc[0] = '<' + first_word_st + '>'
df = df[['start_time', 'end_time', 'word']]

# apply segmentation
full_text = ' '.join(df["word"]) # concatenate all the words
segmenter = DeepSegment('en')
sentence_list = segmenter.segment_long(full_text)
df2 = pd.DataFrame(columns=['sentence'])
df2['sentence'] = sentence_list
df2.index += 1 # index serves as sentence number
df3 = pd.DataFrame(columns=['word_list'])
df3['word_list'] = df2['sentence'].str.split(' ')
s2 = df3.apply(lambda x: pd.Series(x['word_list']), axis=1).stack().reset_index(level=1, drop=True)
s2.name = 'word_2'
df3 = df3.drop('word_list', axis=1).join(s2)
df3['sentence_number'] = df3.index
df3 = df3.reset_index(drop=True)
df3.index += 1
```

Figure 13: Code Snippet for Sentence Segmentation with Timestamps

4.6 Sentence Segmentation with Timestamps

The implementation of sentence segmentation with timestamps is similar to the previous one, except that it has the added functionality of finding the timestamps of the newly-formed sentences after the sentence segmentation. Figure 11 is a code snippet of this implementation. For this implementation, we need to know the start and end times of each word. As a result, the Python script for this implementation works on YouTube transcripts with word-level timestamps. However, the transcripts are required to be cleaned with an existing script first to remove redundant lines and improve the formatting.

4.7 Semantically Similar Formal Sentence Generation with PEGASUS

Using the PEGASUS model, the semantically similar formal sentence generation task is implemented using a Python script that accepts the following inputs:

- CSV file containing query sentences
- Number of similar sentences to be generated per query sentence

For each query sentence in the input CSV file, the Python script generates the specified number of sentences similar to the query sentence using the particular 'tuner007/pegasus_paraphrase' PEGASUS model, which is finetuned for the task of paraphrasing.

4.8 Semantically Similar Formal Sentence Generation with iNLTK

The implementation of semantically similar formal sentence generation with iNLTK is similar to the previous one, except that the iNLTK model is used. The Python script for this implementation has the same inputs as that of the previous implementation.

5 EXPERIMENTATION AND RESULTS

5.1 Experimentation Setup

The two deep-learning models, PEGASUS and iNLTK, are experimented on conversational sentences to evaluate their semantic similarity performance based on the task of generating semantically similar formal sentences. For each conversational sentence, which is also known as a query sentence in this case, the models were set to generate 5 similar sentences. The process of sentence embedding is done on both the query sentences and the generated sentences using the 'bert-base-nli-mean-tokens' Sentence-BERT model. The vector generated for each sentence represents the semantic meaning of the sentence itself. For each query sentence, the cosine similarity score is calculated between its vector and each of the 5 similar sentences' vectors. Cosine similarity is a measure of similarity between two non-zero vectors that measure the cosine of the angle between them. Hence, the cosine similarity metric measures the extent of semantic similarity between a query sentence and each of the 5 generated similar sentences.

Sentence-BERT [19] is an adaptation of the BERT [20] model, with an additional pooling layer implemented. The advantage of the pooling layer is to ensure that the sentence embeddings generated are of fixed size regardless of the varying lengths of the input sentences. Thus, the output vector of any sentence has a fixed number of 768 dimensions. Due to the fact that Sentence-BERT is efficient to a large extent computation-wise

when compared with other sentence embedding models, it is used for the purpose of sentence encoding in this experimentation.

5.2 Dataset Preparation

The conversational text from 240 YouTube videos was obtained in the form of transcripts. The 240 YouTube videos originated from various YouTube channels, with the majority of the videos belonging to medical and health domains. The text obtained from these videos was then made to undergo pre-processing and normalisation, before undergoing sentence segmentation using DeepSegment. 16,318 sentences were acquired after the sentence segmentation process, and 5,500 sentences were randomly sampled from them. These 5,500 query sentences were then used to generate similar sentences using both the PEGASUS and iNLTK models.

5.3 Results and Evaluation

query_sentence	similar_sentence	query_sentence_vector	similar_sentence_vector	cosine_similarity
this feels like a very uh regular afternoon pr...	it feels like they are trying to sell somethin...	[-2.95707226e-01 2.66511977e-01 1.47946370e+...	[4.05243367e-01 5.67617834e-01 2.02121997e+...	0.616032
	it's almost like they are door-to-door sales p...	[-2.95707226e-01 2.66511977e-01 1.47946370e+...	[3.50658387e-01 4.99756306e-01 1.90412211e+...	0.638872
	it's almost like they are door-to-door sales p...	[-2.95707226e-01 2.66511977e-01 1.47946370e+...	[3.00471127e-01 4.40358132e-01 1.81990063e+...	0.623839
	it's almost like they're door-to-door sales pe...	[-2.95707226e-01 2.66511977e-01 1.47946370e+...	[3.35645080e-01 5.50548315e-01 1.87771463e+...	0.631703
	it's almost like they are door-to-door sales p...	[-2.95707226e-01 2.66511977e-01 1.47946370e+...	[2.85724312e-01 4.75420147e-01 1.86791146e+...	0.633522
what did the experiment show before and after ...	there were interesting results that came out o...	[4.15288806e-02 -2.26401985e-01 6.70216799e-...	[-3.62594500e-02 -5.18777192e-01 1.30934268e-...	0.831543
	there were interesting results that came out o...	[4.15288806e-02 -2.26401985e-01 6.70216799e-...	[2.83676058e-01 -7.47129142e-01 5.20087004e-...	0.711926
	there were interesting results from the experi...	[4.15288806e-02 -2.26401985e-01 6.70216799e-...	[8.64474624e-02 -2.67577887e-01 2.79442072e-...	0.840052
	there were interesting results from the experi...	[4.15288806e-02 -2.26401985e-01 6.70216799e-...	[3.88235413e-02 -2.25693837e-01 3.48688424e-...	0.838895
	the results of the experiment came out of the ...	[4.15288806e-02 -2.26401985e-01 6.70216799e-...	[8.25880617e-02 2.78155893e-01 3.53922307e-...	0.834014

Figure 14: Cosine Similarity Scores between Query Sentences and Similar Sentences

Figure 13 shows the cosine similarity score results obtained between each query sentence and the 5 similar sentences.

Model	Mean of Mean Cosine Similarity Scores of Query Sentences
PEGASUS	0.901
iNLTK	0.875

Table 4: Mean of Mean Cosine Similarity Scores of Query Sentences per Model

For each query sentence, we take the mean of the 5 cosine similarity scores that it has with its 5 similar sentences. Then, we take the mean of all the mean cosine similarity scores of the query sentences. We do this for each of the PEGASUS and iNLTK models to get the results in Table 4.

The results for the models show that they are both efficient in terms of ensuring high semantic similarity for the task of similar sentence

generation. However, within the two models, PEGASUS has performed significantly better.

The PEGASUS model has the structure of a encoder-decoder model. Due to the nature of its architecture, it is able to map the input sequences to the output ones, both of which are of variable lengths. On the other hand, the iNLTK model follows a transfer learning algorithm, whereby it focuses on paraphrasing the input sentence by replacing words with those that have the most similar encoding from the language model. Among the paraphrased sentences, the most similar ones to the original input sentence are selected. Furthermore, the PEGASUS model looks at only selected novel words in the input sentence altogether, including them in the output sentence and building the rest of the output sentence. Conversely, the iNLTK model looks at the input sentence tokens one by one to see if they should be replaced and the extent of replacement is dependent on the degree of sentence augmentation provided by the model. Hence, the better result for the PEGASUS model can be attributed to its more robust architecture and the capture of semantics on a more holistic level.

actually he did uh achieve something through this dream previously when he was away	he achieved something through this dream when he was away.
	when he was away, he achieved something through this dream.
	when he was away, he did achieve something through this dream.
	he did achieve something through this dream when he was away.
	he achieved something in this dream when he was away.

Figure 15: Query Sentence and Similar Sentences Generated by PEGASUS Model

actually he did uh achieve something through this dream previously when he was away	still he did uh achieve something through here dream previously if he were away
	actually he did uh achieve anything through these dream formerly when they was away
	actually he did uh achieve anything through this dreams previously during they was away
	still we did uh achieve something through this dream previously if we was away
	actually his did uh achieve something through it hallucination previously when he was out

Figure 16: Query Sentence and Similar Sentences Generated by iNLTK Model

Figures 14 and 15 show one of the query sentences and the 5 similar sentences generated for it by the PEGASUS and iNLTK models respectively. As seen in Figure 14, the filler word 'uh' which is present in the query sentence, is removed in the similar sentences generated. On the other hand, in Figure 15, it is not removed in the similar sentences generated. This shows that the PEGASUS model is able to omit the unique attributes of conversational text such as fillers. Hence, concerning the task of generating formal sentences, the PEGASUS model does it efficiently.

From this experimentation, we can see that the PEGASUS model has performed holistically better in both attributes of the experimentation aim, which is semantic similarity between the query sentences and generated similar sentences, and generating formal sentences from conversational ones.

6 CONCLUSION AND FUTURE WORK

6.1 Summary of Achievements

In this project, a solution to generate semantically similar formal sentences from conversational ones was designed using the PEGASUS model, and analysis and evaluation of deep-learning models on semantically similar formal sentence generation was performed. The use of PEGASUS in generating semantically similar formal sentences has shown that it effectively captures the original semantic meaning of conversational sentences, while being able to mostly filter out conversational text attributes such as fillers, making the generated sentences have a formal nature to a large extent. Furthermore, the implementation of the sentence segmentation on unpunctuated text using DeepSegment has greatly improved the quality of the segmented sentences, in terms of both sensible sentence structure and appropriate length. The multiple text and file processing implementations can act as standardised processes for the input to the language and accoustic models used by ASR systems.

Human attributes, preferences and abilities have been constantly evolving, and ASR systems also have to continuously change and adapt to the former in order to be used effectively. The use of speech recognition has not only grown recently, but has also blended into many other fields that were unimaginable a few decades ago, such as the medical and retail sectors. Even though we have yet to achieve the milestone of ASR systems

holistically understanding the nuances of human speech, the progressive rate of artificial intelligence today is very promising.

6.2 Future Directions

6.2.1 *Enhancing Scalability of Semantically Similar Formal Sentence Generation using PEGASUS*

Considering a huge number of sentences would be required to undergo the process of generating semantically similar formal sentences for purposes such as training ASR systems with sufficiently large amount of data, the task could be implemented to run on a Graphics Processing Unit (GPU) to improve time efficiency greatly. This is because using the above-mentioned PEGASUS model, for each sentence it variably takes about a few seconds (less than five seconds) to generate five similar sentences.

6.2.2 *Finetuning PEGASUS Model*

To be more adapted to the conversational nature of the input text, the PEGASUS model can be finetuned on downstream datasets, customised with conversational sentence embeddings. Optimizers with weight decay such as AdamW [21] and AdaFactor [22] can help in analysing the loss function for the model, which will help to examine how well the model fits to the conversational text data. Experimenting with different values of the learning rate can also be done to allow exploration of the optimal speed of model adaption to the data.

6.2.3 Further Processing of Conversational Text

Additional steps could be taken in pre-processing the conversational text. Slang could be mapped to formal terms with equivalent semantics, whereas fillers can be appropriately recognised and removed. The pre-processing steps may also have the need to evolve such that it can cater to Singapore's sociolinguistics which may transform along with its changing culture. One possibility of this is that new slang words and terminologies specific to the Singaporean context may be used in the future.

A LINK TO PROJECT IMPLEMENTATION FILES

Following is the link to the GitHub repository hosting the project implementation scripts and the relevant documentation: https://github.com/shinygs/FYP-Conversational-Speech.git

REFERENCES

[1] Google, "Voice search mobile use statistics," 2018. [Online]. Available: https://www.thinkwithgoogle.com/marketing-strategies/search/voice-search-mobile-use-statistics/

[2] V. Young and A. Mihailidis, "Difficulties in automatic speech recognition of dysarthric speakers and implications for speech-based applications used by the elderly: A literature review," *Assistive Technology*, vol. 22, no. 2, pp. 99–112, 2010.

[3] M. Djellab, A. Amrouche, A. Bouridane, and N. Mehallegue, "Algerian modern colloquial arabic speech corpus (amcasc): regional accents recognition within complex socio-linguistic environments," *Language Resources and Evaluation*, vol. 51, no. 3, pp. 613–641, 2017.

[4] H. Su, X. Shen, S. Zhao, X. Zhou, P. Hu, R. Zhong, C. Niu, and J. Zhou, "Diversifying dialogue generation with non-conversational text," *arXiv preprint arXiv:2005.04346*, 2020.

[5] M. Baroni and M. Ueyama, "Building general-and special-purpose corpora by web crawling," in *Proceedings of the 13th NIJL international symposium, language corpora: Their compilation and application*, 2006, pp. 31–40.

[6] A. Islam and D. Inkpen, "Semantic similarity of short texts," *Recent Advances in Natural Language Processing V*, vol. 309, pp. 227–236, 2009.

[7] S. Enarvi, P. Smit, S. Virpioja, and M. Kurimo, "Automatic speech recognition with very large conversational finnish and estonian vocabularies," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 11, pp. 2085–2097, 2017.

[8] "youtube-transcript-api: This is an python API which allows you to get the transcripts/subtitles for a given YouTube video. It also works for automatically generated subtitles, supports translating subtitles and it does not require a headless browser, like other selenium based solutions do!" [Online]. Available: https://github.com/jdepoix/youtube-transcript-api

[9] "youtube-dl." [Online]. Available: https://youtube-dl.org/

[10] "Subtitle Edit." [Online]. Available: https://nikse.dk/SubtitleEdit/

[11] "deepsegment: Sentence Segmentation with sequece tagging." [Online]. Available: https://github.com/bedapudi6788/Deep-Segmentation

[12] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," 2019.

[13] G. Arora, "iNLTK: Natural language toolkit for indic languages," in *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 66–71. [Online]. Available: https://www.aclweb.org/anthology/2020.nlposs-1.10

[14] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[15] "pydub: Manipulate audio with an simple and easy high level interface." [Online]. Available: http://pydub.com

[16] "FFmpeg." [Online]. Available: https://ffmpeg.org/

[17] "PyTorch." [Online]. Available: https://www.pytorch.org

[18] "PRAW: The Python Reddit API Wrapper — PRAW 7.5.0 documentation." [Online]. Available: https://praw.readthedocs.io/en/stable/

[19] "SentenceTransformers Documentation — Sentence-Transformers documentation." [Online]. Available: https://www.sbert.net/

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[21] "AdamW — PyTorch 1.11.0 documentation." [Online]. Available: https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html

[22] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4596–4604.