NLP Assignment 1

August 31, 2024

Introduction

Problem Statement

To perform exploratory text analysis on the provided transcript dataset to service meaningful insights.

Analysis Aim and Objectives:

The goal of this analysis is to determine whether the type of radio station and its intended audience location will cause any side effects within the content discussed that can be located within the transcripts.

Furthermore, differences between national radio and commercial radio will be explored, to determine if there are any major differences between national radio and commercial radio in regards to what language they use.

Preprocessing

Overview

The dataset contains a collection of 29 plain text files that are transcripts of Australian Radio Talk. 14 Transcribed recordings of the talkback are from National Radio broadcasts such as ABC National Radio, ABC Radio Broadcasts to Eastern Australia, Southern Australia and Western Australia. There are also 15 transcribed recordings that are from commercial stations broadcasted to Eastern, Southern and Western Australia.

[85]:
```
import os
import pandas as pd
from collections import Counter
from wordcloud import WordCloud
import matplotlib.pyplot as plt

folder_path = 'AT1 dataset_AusRadioTalkback'
data = []


for filename in os.listdir(folder_path):
    if filename.endswith(".txt"):
        file_path = os.path.join(folder_path, filename)
        with open(file_path, 'r', encoding='utf-8') as file:
            content = file.read()
```

```python
            data.append({'filename': filename, 'content': content})


df = pd.DataFrame(data)
print(df)
```

```
                     filename  \
0   1_Dataset Description.txt   
1             ABCE1-plain.txt   
2             ABCE2-plain.txt   
3             ABCE3-plain.txt   
4             ABCE4-plain.txt   
5            ABCNE1-plain.txt   
6            ABCNE2-plain.txt   
7             COME1-plain.txt   
8             COME2-plain.txt   
9             COME3-plain.txt   
10            COME4-plain.txt   
11            COME5-plain.txt   
12            COME6-plain.txt   
13            COME7-plain.txt   
14            COME8-plain.txt   
15           COMNE1-plain.txt   
16           COMNE2-plain.txt   
17           COMNE3-plain.txt   
18           COMNE4-plain.txt   
19           COMNE5-plain.txt   
20           COMNE6-plain.txt   
21           COMNE7-plain.txt   
22             NAT1-plain.txt   
23             NAT2-plain.txt   
24             NAT3-plain.txt   
25             NAT4-plain.txt   
26             NAT5-plain.txt   
27             NAT6-plain.txt   
28             NAT7-plain.txt   
29             NAT8-plain.txt   

                                              content  
0   Name\n\nAustralian Radio Talkback\n\nDescripti…  
1    Thanks for that John Hall now John Hall will …  
2    Ah look l Les Pete.\n.\n Simon.\n G'day Peto…  
3    If you haven't been with us before this how i…  
4    Uh blue-tongues'd be  unlikely to eat them be…  
5    A very good afternoon to you Roly.\n Good aft…  
6    And Greg Kerrin is my guest. Hello Greg.\n G'…  
7    Good morning and welcome to another Two G B w…  
8    Good morning everyone and welcome to a very f…  
```

```
9     The doctor is in the lines are open one-three…
10    Morning Mark.\n\n Uh uh good morning John. Um…
11    Here's Sharina's Saturday Nights the positive…
12    Where we are talking about how long it takes …
13    Program G P Dr Sally Cockburn good morning.\n…
14    Mix one-oh-six-point-five We Did It All For L…
15    Freo Dockers skipper there Peter Bell hello e…
16    Good afternoon Howard Sattler with you welcom…
17    We are Talking Real Estate on eight-eighty-tw…
18    Thank you Len and a very good morning to a ra…
19    Lawyer Bob on the job merry Christmas.\n Merr…
20    Well the wait ended for year twelve students …
21    Been to the city lately tried to find a park …
22    One-eight-hundred-eight-oh-two-three-four-one…
23    And speaking of welcomes and people we love R…
24    Good morning how are you.\n Ih is it true you…
25    Eighteen minutes past ten eighteen past nine …
26    So now it's welcome first to our expert panel…
27    you're with Mel in the morning it's fourteen …
28    Five A M in New York hey. There's gotta be  s…
29    Hello and welcome to the Chatroom with Gaby t…
```

Cleaning

To start, all code was loaded into a dataframe and set up with a text processing function. The order of the preprocessing is as follows:

1. Spaces were all added after parenthesis to create consistency for tokenizing.
2. Hyphens were all converted to spaces, to allow tokenization to be done correctly. Without this, hyphens were getting removed and whenever they were used, words were being combined together without spaces separating them.
3. Punctuation was removed. This includes: Quotations, eclipses, quotes and hyphens.
4. All the text was turned to lowercase for text consistency.
5. All stop words were removed. Multiple tests with different stopwords were done. This included removing all common words that didn't provide any context, removing no words, removing strange artefacts of the tokenisation such as "c", "ih", "n".
6. Finally all words were turned into their stem forms using the nltk.stem library's built in functions.

[87]:
```python
#Modified code from Lab 2 Part 2
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import re
import string
from nltk import pos_tag
from nltk.util import bigrams, trigrams
```

```python
# Download necessary NLTK data
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')

class TextPreprocessor:
    def __init__(self, custom_punctuation=None, custom_stopwords=None):
        self.punctuation = string.punctuation
        if custom_punctuation:
            self.punctuation += custom_punctuation

        self.stop_words = set(stopwords.words('english'))
        if custom_stopwords:
            self.stop_words.update(custom_stopwords)

        self.stemmer = PorterStemmer()

    def remove_punctuation(self, text):
        return ''.join([char for char in text if char not in self.punctuation])

    def add_space_after_parenthesis(self, text):
        return re.sub(r'\)', ') ', text)

    def hyphen_to_space(self, text):
        return re.sub(r'\-', ' ', text)

    def to_lowercase(self, text):
        return text.lower()

    def remove_stopwords(self, text):
        words = word_tokenize(text)
        return ' '.join([word for word in words if word not in self.stop_words])

    def stem_words(self, text):
        words = word_tokenize(text)
        return ' '.join([self.stemmer.stem(word) for word in words])

    # https://www.nltk.org/book/ch05.html nouns tags found here
    def keep_pos_tag(self, text, tag): #tag is a dictionary eg {"NN", NNP"}
        words = word_tokenize(text)
        return ' '.join([word for word, pos in pos_tag(words) if pos in tag])


    #Order matters - how you call these methods is how the text will be
 ↪processed step-by-step
    def preprocess(self, text):
        text = self.add_space_after_parenthesis(text)
```

```
        text = self.hyphen_to_space(text)
        text = self.remove_punctuation(text)
        text = self.to_lowercase(text)
        text = self.remove_stopwords(text)
        text = self.stem_words(text)
        return text


    def preprocess_keep_pos_tag(self, text, tag):
        text = self.add_space_after_parenthesis(text)
        text = self.hyphen_to_space(text)
        text = self.remove_punctuation(text)
        text = self.to_lowercase(text)
        text = self.remove_stopwords(text)
        text = self.keep_pos_tag(text, tag)
        text = self.stem_words(text)
        return text
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\mason\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\mason\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\mason\AppData\Roaming\nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

[88]:
```
print(df)
```

```
                     filename  \
0   1_Dataset Description.txt
1             ABCE1-plain.txt
2             ABCE2-plain.txt
3             ABCE3-plain.txt
4             ABCE4-plain.txt
5            ABCNE1-plain.txt
6            ABCNE2-plain.txt
7             COME1-plain.txt
8             COME2-plain.txt
9             COME3-plain.txt
10            COME4-plain.txt
11            COME5-plain.txt
12            COME6-plain.txt
13            COME7-plain.txt
14            COME8-plain.txt
15           COMNE1-plain.txt
```

```
16          COMNE2-plain.txt
17          COMNE3-plain.txt
18          COMNE4-plain.txt
19          COMNE5-plain.txt
20          COMNE6-plain.txt
21          COMNE7-plain.txt
22            NAT1-plain.txt
23            NAT2-plain.txt
24            NAT3-plain.txt
25            NAT4-plain.txt
26            NAT5-plain.txt
27            NAT6-plain.txt
28            NAT7-plain.txt
29            NAT8-plain.txt

                                                content
0   Name\n\nAustralian Radio Talkback\n\nDescripti…
1    Thanks for that John Hall now John Hall will …
2    Ah look l Les Pete.\n.\n Simon.\n G'day Peto…
3    If you haven't been with us before this how i…
4    Uh blue-tongues'd be  unlikely to eat them be…
5    A very good afternoon to you Roly.\n Good aft…
6    And Greg Kerrin is my guest. Hello Greg.\n G'…
7    Good morning and welcome to another Two G B w…
8    Good morning everyone and welcome to a very f…
9    The doctor is in the lines are open one-three…
10   Morning Mark.\n\n Uh uh good morning John. Um…
11   Here's Sharina's Saturday Nights the positive…
12   Where we are talking about how long it takes …
13   Program G P Dr Sally Cockburn good morning.\n…
14   Mix one-oh-six-point-five We Did It All For L…
15   Freo Dockers skipper there Peter Bell hello e…
16   Good afternoon Howard Sattler with you welcom…
17   We are Talking Real Estate on eight-eighty-tw…
18   Thank you Len and a very good morning to a ra…
19   Lawyer Bob on the job merry Christmas.\n Merr…
20   Well the wait ended for year twelve students …
21   Been to the city lately tried to find a park …
22   One-eight-hundred-eight-oh-two-three-four-one…
23   And speaking of welcomes and people we love R…
24   Good morning how are you.\n Ih is it true you…
25   Eighteen minutes past ten eighteen past nine …
26   So now it's welcome first to our expert panel…
27   you're with Mel in the morning it's fourteen …
28   Five A M in New York hey. There's gotta be  s…
29   Hello and welcome to the Chatroom with Gaby t…
```

[89]:
```
#Remove first entry (the dataset description)
df = df.iloc[1:]
```

[90]:
```
df
```

[90]:
```
            filename                                            content
1     ABCE1-plain.txt  Thanks for that John Hall now John Hall will …
2     ABCE2-plain.txt  Ah look l Les Pete.\n.\n Simon.\n G'day Peto…
3     ABCE3-plain.txt  If you haven't been with us before this how i…
4     ABCE4-plain.txt  Uh blue-tongues'd be  unlikely to eat them be…
5    ABCNE1-plain.txt  A very good afternoon to you Roly.\n Good aft…
6    ABCNE2-plain.txt  And Greg Kerrin is my guest. Hello Greg.\n G'…
7     COME1-plain.txt  Good morning and welcome to another Two G B w…
8     COME2-plain.txt  Good morning everyone and welcome to a very f…
9     COME3-plain.txt  The doctor is in the lines are open one-three…
10    COME4-plain.txt  Morning Mark.\n\n Uh uh good morning John. Um…
11    COME5-plain.txt  Here's Sharina's Saturday Nights the positive…
12    COME6-plain.txt  Where we are talking about how long it takes …
13    COME7-plain.txt  Program G P Dr Sally Cockburn good morning.\n…
14    COME8-plain.txt  Mix one-oh-six-point-five We Did It All For L…
15   COMNE1-plain.txt  Freo Dockers skipper there Peter Bell hello e…
16   COMNE2-plain.txt  Good afternoon Howard Sattler with you welcom…
17   COMNE3-plain.txt  We are Talking Real Estate on eight-eighty-tw…
18   COMNE4-plain.txt  Thank you Len and a very good morning to a ra…
19   COMNE5-plain.txt  Lawyer Bob on the job merry Christmas.\n Merr…
20   COMNE6-plain.txt  Well the wait ended for year twelve students …
21   COMNE7-plain.txt  Been to the city lately tried to find a park …
22     NAT1-plain.txt  One-eight-hundred-eight-oh-two-three-four-one…
23     NAT2-plain.txt  And speaking of welcomes and people we love R…
24     NAT3-plain.txt  Good morning how are you.\n Ih is it true you…
25     NAT4-plain.txt  Eighteen minutes past ten eighteen past nine …
26     NAT5-plain.txt  So now it's welcome first to our expert panel…
27     NAT6-plain.txt  you're with Mel in the morning it's fourteen …
28     NAT7-plain.txt  Five A M in New York hey. There's gotta be  s…
29     NAT8-plain.txt  Hello and welcome to the Chatroom with Gaby t…
```

[91]:
```
#Code that combines all the different df rows based on filename into 1 long␣
↪string grouped by radio Station

#Create prefix column using regex (take characters from beginning of filename,␣
↪until you hit a number)
df['prefix'] = df['filename'].apply(lambda x: re.match(r'^[A-Z]+', x).group())

#Group by prefix
df_combined = df.groupby('prefix')['content'].apply(' '.join).reset_index()
```

[92]:
```
df_combined
```

[92]:

	prefix	content
0	ABCE	Thanks for that John Hall now John Hall will …
1	ABCNE	A very good afternoon to you Roly.\n Good aft…
2	COME	Good morning and welcome to another Two G B w…
3	COMNE	Freo Dockers skipper there Peter Bell hello e…
4	NAT	One-eight-hundred-eight-oh-two-three-four-one…

[93]:
```
# Get word frequencies
def get_word_frequencies(text, top_n=10):
    words = word_tokenize(text)
    word_freq = Counter(words)
    return word_freq.most_common(top_n)

def get_word_freq_percentage(text, top_n=20):
    words = word_tokenize(text)
    word_freq = Counter(words)
    return [(word, round((count / len(words) * 100), 2)) for word, count in␣
↪word_freq.most_common(top_n)]
```

[94]:
```
#testing without stop words
custom_punctuation = "''""…""...""``" '"' '-'  # Add any custom punctuation␣
↪marks here
custom_stopwords = ["n", "c", "ih", "w"]
preprocessor = TextPreprocessor(custom_punctuation, custom_stopwords)
word_freqs = {}

for index, row in df_combined.iterrows():
    processed = preprocessor.preprocess(row['content'])
    word_freqs[row['prefix']] = get_word_frequencies(processed, top_n=50)

# Combine word frequencies into a DataFrame
df_word_freqs = pd.DataFrame(word_freqs).fillna(0)
```

[95]:
```
df_word_freqs
```

[95]:

	ABCE	ABCNE	COME	COMNE \
0	(uh, 580)	(uh, 179)	(uh, 1207)	(uh, 1259)
1	(um, 286)	(um, 68)	(um, 647)	(um, 512)
2	(yeah, 232)	(one, 66)	(well, 644)	(yeah, 365)
3	(well, 200)	(okay, 56)	(yeah, 466)	(well, 332)
4	(like, 167)	(well, 50)	(got, 435)	(one, 299)
5	(ye, 166)	(ye, 50)	(like, 433)	(that, 291)
6	(that, 146)	(that, 48)	(good, 423)	(go, 278)
7	(get, 139)	(good, 47)	(oh, 412)	(got, 273)
8	(oh, 133)	(got, 47)	(that, 409)	(get, 264)
9	(think, 130)	(like, 41)	(go, 396)	(like, 230)
10	(one, 123)	(oh, 41)	(get, 374)	(good, 227)

11	(good, 118)	(yeah, 39)	(think, 368)	(think, 225)
12	(got, 111)	(word, 39)	(your, 348)	(look, 218)
13	(go, 109)	(thank, 39)	(okay, 339)	(yknow, 207)
14	(would, 99)	(right, 36)	(im, 337)	(year, 204)
15	(realli, 95)	(year, 35)	(realli, 326)	(say, 194)
16	(sort, 94)	(get, 34)	(know, 325)	(two, 189)
17	(look, 85)	(dont, 33)	(one, 307)	(oh, 166)
18	(thank, 84)	(theyr, 33)	(ye, 273)	(im, 164)
19	(theyr, 82)	(bit, 32)	(dont, 259)	(dont, 158)
20	(im, 77)	(use, 30)	(look, 254)	(theyr, 150)
21	(know, 76)	(afternoon, 26)	(thing, 244)	(thing, 147)
22	(two, 74)	(alright, 26)	(say, 234)	(there, 143)
23	(dont, 73)	(mm, 25)	(time, 223)	(know, 140)
24	(say, 71)	(littl, 24)	(two, 215)	(time, 138)
25	(okay, 70)	(time, 24)	(would, 210)	(thank, 134)
26	(book, 70)	(your, 24)	(ive, 206)	(peopl, 133)
27	(want, 69)	(hundr, 24)	(thank, 200)	(mean, 129)
28	(see, 66)	(flower, 24)	(youv, 200)	(ive, 122)
29	(bit, 66)	(think, 23)	(love, 192)	(okay, 118)
30	(actual, 66)	(hyphen, 23)	(year, 190)	(your, 112)
31	(mm, 62)	(thing, 22)	(back, 181)	(bit, 106)
32	(id, 61)	(call, 22)	(there, 180)	(would, 105)
33	(yknow, 59)	(know, 22)	(three, 175)	(need, 103)
34	(thing, 59)	(tilli, 22)	(call, 171)	(realli, 101)
35	(right, 56)	(sprog, 21)	(peopl, 165)	(much, 99)
36	(there, 56)	(six, 21)	(morn, 164)	(want, 94)
37	(use, 55)	(ive, 20)	(sort, 164)	(right, 92)
38	(littl, 53)	(come, 20)	(yknow, 163)	(three, 92)
39	(time, 53)	(take, 19)	(lot, 159)	(back, 90)
40	(ive, 53)	(mean, 19)	(see, 158)	(see, 90)
41	(plant, 51)	(three, 19)	(right, 154)	(ye, 89)
42	(put, 49)	(there, 19)	(take, 150)	(make, 88)
43	(year, 49)	(im, 19)	(theyr, 149)	(twenti, 87)
44	(call, 48)	(cut, 19)	(number, 147)	(put, 87)
45	(come, 48)	(much, 19)	(littl, 147)	(come, 86)
46	(thought, 48)	(go, 19)	(bit, 144)	(doubl, 85)
47	(your, 48)	(roli, 18)	(want, 143)	(youv, 84)
48	(take, 47)	(two, 18)	(come, 142)	(four, 83)
49	(youv, 47)	(hello, 17)	(mean, 134)	(eight, 80)

	NAT
0	(uh, 1877)
1	(um, 1186)
2	(think, 553)
3	(yknow, 516)
4	(well, 400)
5	(like, 398)

```
6     (yeah, 383)
7    (peopl, 379)
8      (one, 350)
9       (go, 349)
10    (that, 308)
11     (say, 305)
12      (im, 304)
13      (oh, 298)
14    (good, 246)
15    (dont, 246)
16     (got, 239)
17      (ye, 223)
18    (know, 221)
19   (thing, 217)
20   (would, 217)
21     (get, 215)
22    (mean, 209)
23   (thank, 202)
24    (look, 200)
25    (book, 196)
26   (theyr, 181)
27  (realli, 179)
28   (there, 175)
29    (come, 170)
30    (time, 167)
31    (year, 166)
32     (ive, 161)
33    (call, 154)
34  (actual, 153)
35    (sort, 153)
36    (much, 152)
37    (talk, 148)
38    (your, 147)
39    (read, 144)
40     (way, 134)
41     (two, 132)
42      (us, 127)
43   (right, 124)
44    (said, 124)
45    (need, 124)
46     (see, 124)
47    (okay, 123)
48     (let, 122)
49    (take, 121)
```

Exploratory Data Analysis

Starting with common words analysis (Minimal Stop Words used):

Above is a list of all most frequent words separated by the different radio stations. The most common words across all radio stations are the same, with filler words such as “uh” and “um” substantially beating out the rest of the word counts significantly. Despite the extremely large amount of use of these filler words, they don’t contribute anything significant (ranging from 2.98% to 4.41% of the total word count) to the total distribution of the words within the transcriptions.

From an initial glance, standouts include “yknow” which is a common Australian shorthand for “you know” and is an extremely common use of colloquial language in Australia in general. From the word count table, we can see that it is the **4th** most common word for the National Australia Radio (NAT), ranked **14th** for commercial stations broadcasting to the Southern and Western side of Australia (COMNE) and Ranked **39th** for Commercial Eastern Australian Radio (COME). Finally it is ranked **34th** for Eastern ABC Radio (ABCE) and does not appear within the top 20 for Southern and Western Australia. This use of colloquial language is further enforced with the use of “gunna” which can be seen in the word cloud for NAT that consists of only nouns (see next page).

As “yknow” and “gunna” is considered colloquial language, it is commonly used to allow the listener to relate as “it’s the same language as what the listener may use themselves”, this allows the listener to become more invested in the chatter as it sounds like a conversation between a group of friends and this mentality allows the radio cast to benefit from more relatability and hence have more listeners. As the ABC National Radio is Australia-wide, this could be a side affect of having a more diverse cast talking on the radio, with people from all backgrounds both professional and casual, resulting in more inclusive and casual language used to appeal to as wide an audience as possible.

[97]:
```
#Lets get the word frequency as a % of total words
word_freqs_percentage = {}
for index, row in df_combined.iterrows():
    processed = preprocessor.preprocess(row['content'])
    word_freqs_percentage[row['prefix']] = get_word_freq_percentage(processed,␣
 ↪top_n=50)

# Combine word frequencies into a DataFrame
df_word_freq_percentage = pd.DataFrame(word_freqs_percentage).fillna(0)
```

[98]:
```
df_word_freq_percentage
```

[98]:
```
              ABCE            ABCNE             COME            COMNE  \
0       (uh, 3.79)       (uh, 3.68)       (uh, 2.98)       (uh, 4.41)
1       (um, 1.87)        (um, 1.4)       (um, 1.59)       (um, 1.79)
2     (yeah, 1.52)      (one, 1.36)     (well, 1.59)     (yeah, 1.28)
3     (well, 1.31)     (okay, 1.15)     (yeah, 1.15)     (well, 1.16)
4     (like, 1.09)     (well, 1.03)      (got, 1.07)      (one, 1.05)
5       (ye, 1.08)       (ye, 1.03)     (like, 1.07)     (that, 1.02)
6     (that, 0.95)     (that, 0.99)     (good, 1.04)       (go, 0.97)
7      (get, 0.91)     (good, 0.97)       (oh, 1.02)      (got, 0.96)
8       (oh, 0.87)      (got, 0.97)     (that, 1.01)      (get, 0.92)
9    (think, 0.85)     (like, 0.84)       (go, 0.98)       (like, 0.8)
```

10	(one, 0.8)	(oh, 0.84)	(get, 0.92)	(good, 0.79)
11	(good, 0.77)	(yeah, 0.8)	(think, 0.91)	(think, 0.79)
12	(got, 0.73)	(word, 0.8)	(your, 0.86)	(look, 0.76)
13	(go, 0.71)	(thank, 0.8)	(okay, 0.84)	(yknow, 0.72)
14	(would, 0.65)	(right, 0.74)	(im, 0.83)	(year, 0.71)
15	(realli, 0.62)	(year, 0.72)	(realli, 0.8)	(say, 0.68)
16	(sort, 0.61)	(get, 0.7)	(know, 0.8)	(two, 0.66)
17	(look, 0.56)	(dont, 0.68)	(one, 0.76)	(oh, 0.58)
18	(thank, 0.55)	(theyr, 0.68)	(ye, 0.67)	(im, 0.57)
19	(theyr, 0.54)	(bit, 0.66)	(dont, 0.64)	(dont, 0.55)
20	(im, 0.5)	(use, 0.62)	(look, 0.63)	(theyr, 0.52)
21	(know, 0.5)	(afternoon, 0.53)	(thing, 0.6)	(thing, 0.51)
22	(two, 0.48)	(alright, 0.53)	(say, 0.58)	(there, 0.5)
23	(dont, 0.48)	(mm, 0.51)	(time, 0.55)	(know, 0.49)
24	(say, 0.46)	(littl, 0.49)	(two, 0.53)	(time, 0.48)
25	(okay, 0.46)	(time, 0.49)	(would, 0.52)	(thank, 0.47)
26	(book, 0.46)	(your, 0.49)	(ive, 0.51)	(peopl, 0.47)
27	(want, 0.45)	(hundr, 0.49)	(thank, 0.49)	(mean, 0.45)
28	(see, 0.43)	(flower, 0.49)	(youv, 0.49)	(ive, 0.43)
29	(bit, 0.43)	(think, 0.47)	(love, 0.47)	(okay, 0.41)
30	(actual, 0.43)	(hyphen, 0.47)	(year, 0.47)	(your, 0.39)
31	(mm, 0.41)	(thing, 0.45)	(back, 0.45)	(bit, 0.37)
32	(id, 0.4)	(call, 0.45)	(there, 0.44)	(would, 0.37)
33	(yknow, 0.39)	(know, 0.45)	(three, 0.43)	(need, 0.36)
34	(thing, 0.39)	(tilli, 0.45)	(call, 0.42)	(realli, 0.35)
35	(right, 0.37)	(sprog, 0.43)	(peopl, 0.41)	(much, 0.35)
36	(there, 0.37)	(six, 0.43)	(morn, 0.4)	(want, 0.33)
37	(use, 0.36)	(ive, 0.41)	(sort, 0.4)	(right, 0.32)
38	(littl, 0.35)	(come, 0.41)	(yknow, 0.4)	(three, 0.32)
39	(time, 0.35)	(take, 0.39)	(lot, 0.39)	(back, 0.31)
40	(ive, 0.35)	(mean, 0.39)	(see, 0.39)	(see, 0.31)
41	(plant, 0.33)	(three, 0.39)	(right, 0.38)	(ye, 0.31)
42	(put, 0.32)	(there, 0.39)	(take, 0.37)	(make, 0.31)
43	(year, 0.32)	(im, 0.39)	(theyr, 0.37)	(twenti, 0.3)
44	(call, 0.31)	(cut, 0.39)	(number, 0.36)	(put, 0.3)
45	(come, 0.31)	(much, 0.39)	(littl, 0.36)	(come, 0.3)
46	(thought, 0.31)	(go, 0.39)	(bit, 0.35)	(doubl, 0.3)
47	(your, 0.31)	(roli, 0.37)	(want, 0.35)	(youv, 0.29)
48	(take, 0.31)	(two, 0.37)	(come, 0.35)	(four, 0.29)
49	(youv, 0.31)	(hello, 0.35)	(mean, 0.33)	(eight, 0.28)

	NAT
0	(uh, 4.37)
1	(um, 2.76)
2	(think, 1.29)
3	(yknow, 1.2)
4	(well, 0.93)

5	(like, 0.93)
6	(yeah, 0.89)
7	(peopl, 0.88)
8	(one, 0.81)
9	(go, 0.81)
10	(that, 0.72)
11	(say, 0.71)
12	(im, 0.71)
13	(oh, 0.69)
14	(good, 0.57)
15	(dont, 0.57)
16	(got, 0.56)
17	(ye, 0.52)
18	(know, 0.51)
19	(thing, 0.5)
20	(would, 0.5)
21	(get, 0.5)
22	(mean, 0.49)
23	(thank, 0.47)
24	(look, 0.47)
25	(book, 0.46)
26	(theyr, 0.42)
27	(realli, 0.42)
28	(there, 0.41)
29	(come, 0.4)
30	(time, 0.39)
31	(year, 0.39)
32	(ive, 0.37)
33	(call, 0.36)
34	(actual, 0.36)
35	(sort, 0.36)
36	(much, 0.35)
37	(talk, 0.34)
38	(your, 0.34)
39	(read, 0.34)
40	(way, 0.31)
41	(two, 0.31)
42	(us, 0.3)
43	(right, 0.29)
44	(said, 0.29)
45	(need, 0.29)
46	(see, 0.29)
47	(okay, 0.29)
48	(let, 0.28)
49	(take, 0.28)

[99]:
```python
#Comparing Word counts with more stop words
custom_punctuation = "''""…""...""``" '"' '-'  # Add any custom punctuation␣
 ↪marks here
custom_stopwords = ["n", "c", "ih", "w", "uh", "um", "yes", "yeah","yknow",␣
 ↪"well", "like", "okay", "that", "thats", "one", "got", "oh", "think",␣
 ↪"good", "youre", "dont", "im", "get", "really", "theyre", "know", "ive",␣
 ↪"say", "two", "three", "would", "right", "bit", "word", "thank", "hello",␣
 ↪"youve", "sort", "look", "theres", "mm", "four", "five", "six", "seven",␣
 ↪"eight", "nine", "ten"]
preprocessor = TextPreprocessor(custom_punctuation, custom_stopwords)
word_freqs = {}

for index, row in df_combined.iterrows():
    processed = preprocessor.preprocess(row['content'])
    word_freqs[row['prefix']] = get_word_frequencies(processed, top_n=20)

# Combine word frequencies into a DataFrame
df_word_freqs = pd.DataFrame(word_freqs).fillna(0)
```

[100]:
```python
df_word_freqs
```

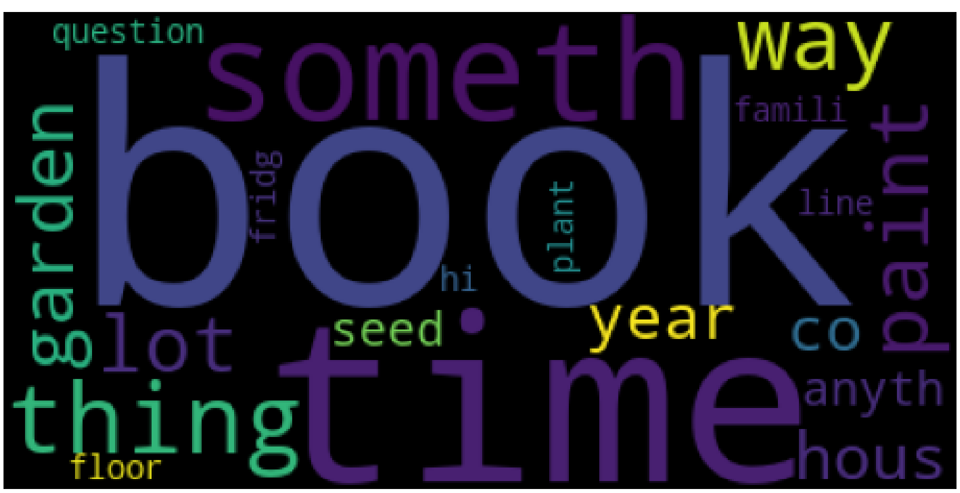
[100]:

	ABCE	ABCNE	COME	COMNE	NAT
0	(go, 109)	(year, 35)	(go, 396)	(go, 278)	(peopl, 379)
1	(book, 70)	(use, 30)	(thing, 244)	(year, 204)	(go, 349)
2	(want, 69)	(afternoon, 26)	(time, 223)	(thing, 147)	(thing, 217)
3	(see, 66)	(alright, 26)	(love, 192)	(time, 138)	(mean, 209)
4	(actual, 66)	(littl, 24)	(year, 190)	(peopl, 133)	(book, 196)
5	(id, 61)	(time, 24)	(back, 181)	(mean, 129)	(come, 170)
6	(thing, 59)	(hundr, 24)	(call, 171)	(need, 103)	(time, 167)
7	(use, 55)	(flower, 24)	(peopl, 165)	(much, 99)	(year, 166)
8	(littl, 53)	(hyphen, 23)	(morn, 164)	(want, 94)	(call, 154)
9	(time, 53)	(thing, 22)	(lot, 159)	(back, 90)	(actual, 153)
10	(plant, 51)	(call, 22)	(see, 158)	(see, 90)	(much, 152)
11	(put, 49)	(tilli, 22)	(take, 150)	(make, 88)	(talk, 148)
12	(year, 49)	(sprog, 21)	(number, 147)	(twenti, 87)	(read, 144)
13	(call, 48)	(come, 20)	(littl, 147)	(put, 87)	(way, 134)
14	(come, 48)	(take, 19)	(want, 143)	(come, 86)	(us, 127)
15	(thought, 48)	(mean, 19)	(come, 142)	(doubl, 85)	(said, 124)
16	(take, 47)	(cut, 19)	(mean, 134)	(he, 77)	(need, 124)
17	(read, 47)	(much, 19)	(much, 125)	(gunna, 76)	(see, 124)
18	(could, 46)	(go, 19)	(put, 118)	(actual, 76)	(let, 122)
19	(yep, 46)	(roli, 18)	(talk, 118)	(littl, 76)	(take, 121)

Above is the same analysis, but with more self-chosen stop words to remove as much of the "meaningless" words from the list

[102]:
```
#Comparing Word counts extracting only nouns
custom_punctuation = "''""…""...""``" '"' '_'
custom_stopwords = ["n", "c", "ih", "w", "uh", "um", "yes", "yeah","yknow",␣
↪"well", "like", "okay", "that", "thats", "one", "got", "oh", "think",␣
↪"good", "youre", "dont", "im", "get", "really", "theyre", "know", "ive",␣
↪"say", "two", "three", "would", "right", "bit", "word", "thank", "hello",␣
↪"youve", "sort", "look", "theres", "mm", "four", "five", "six", "seven",␣
↪"eight", "nine", "ten", "b", "d", "co", "yep", "id"]
preprocessor = TextPreprocessor(custom_punctuation, custom_stopwords)
word_freqs = {}

for index, row in df_combined.iterrows():
    processed = preprocessor.preprocess_keep_pos_tag(row['content'], {"NN"})␣
↪#NN = Noun
    word_freqs[row['prefix']] = get_word_frequencies(processed, top_n=20)

    print(row['prefix'])
    wordcloud = WordCloud(max_words=20).generate(processed)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()

# Combine word frequencies into a DataFrame
df_word_freqs = pd.DataFrame(word_freqs).fillna(0)
```

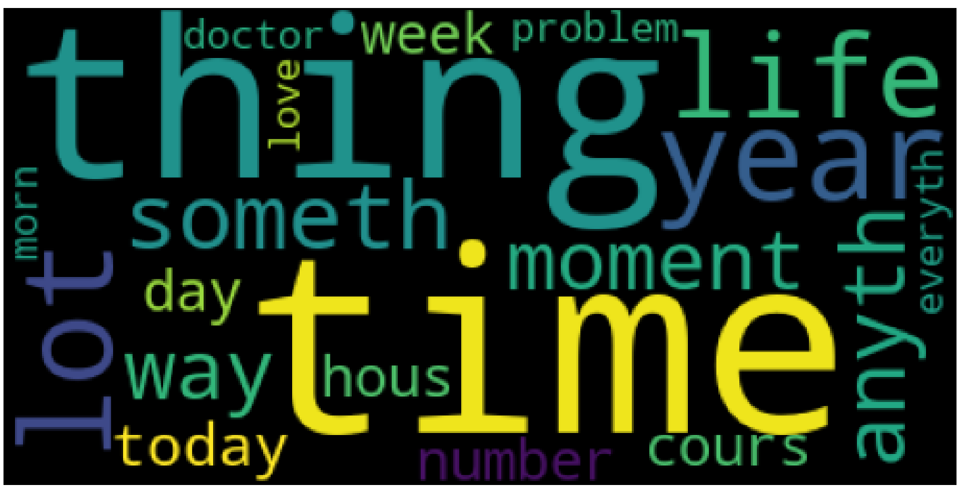
ABCE



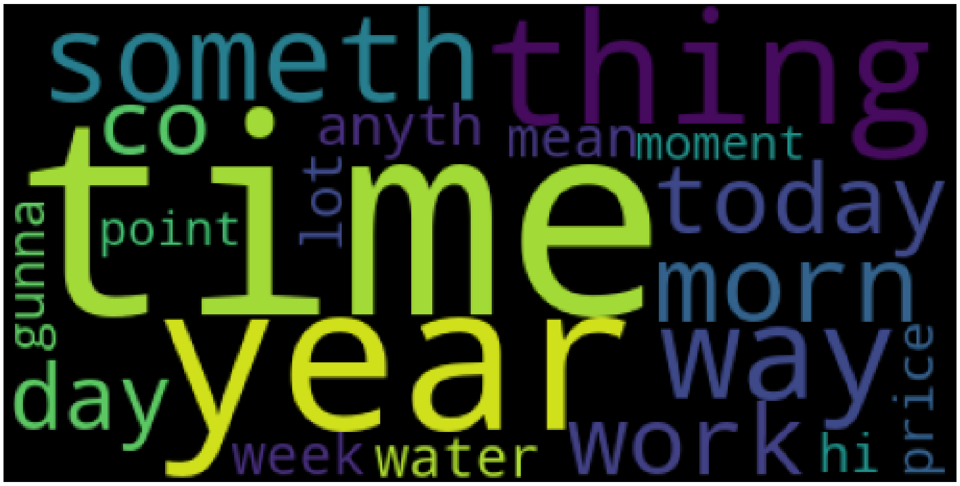
ABCNE


someon
someth
water
year
mean
trevor
john
leon
roli
gday
use
sprog
salt
time
afternoon
thing
way
flower
golf
week

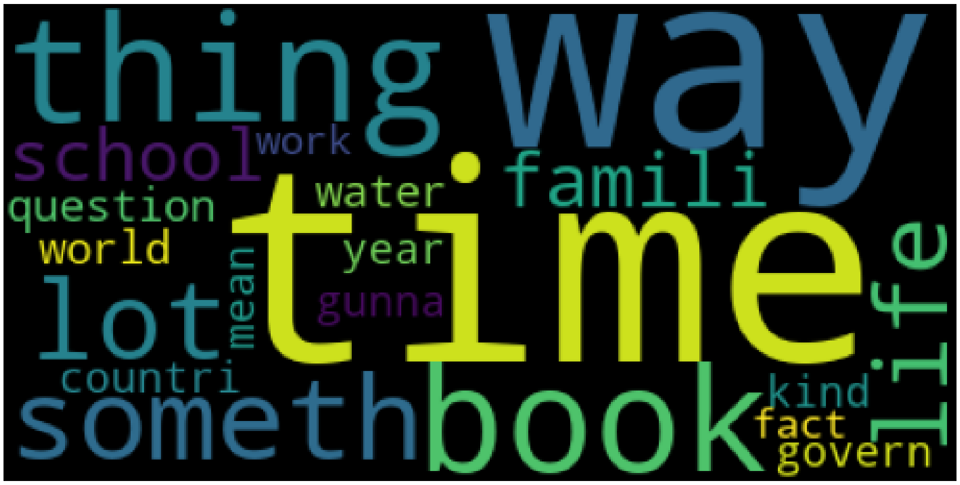
COME


doctor
week
problem
thing
life
love
year
morn
someth
moment
everyth
lot
anyth
day
time
way
hous
today
number
cours

COMNE



NAT



[103]:
```
df_word_freqs
```

[103]:

```
             ABCE              ABCNE            COME            COMNE  \
0      (book, 64)         (year, 27)     (time, 197)     (year, 129)
1      (time, 48)    (afternoon, 26)     (morn, 164)     (time, 121)
2    (someth, 41)         (time, 21)    (thing, 144)     (thing, 85)
3     (thing, 40)         (salt, 13)   (number, 109)      (morn, 66)
4       (way, 33)       (someon, 13)     (year, 101)       (way, 60)
5     (paint, 31)         (roli, 12)      (lot, 101)    (someth, 54)
6    (garden, 29)       (someth, 12)      (life, 91)     (today, 41)
```

7	(lot, 25)	(water, 11)	(someth, 87)	(work, 41)
8	(hous, 24)	(gday, 10)	(day, 80)	(co, 39)
9	(co, 23)	(way, 10)	(anyth, 78)	(day, 38)
10	(year, 23)	(number, 10)	(way, 70)	(twenti, 38)
11	(anyth, 21)	(flower, 10)	(moment, 69)	(anyth, 37)
12	(seed, 21)	(john, 9)	(today, 68)	(lot, 37)
13	(question, 21)	(thing, 9)	(love, 63)	(week, 35)
14	(fridg, 21)	(week, 9)	(hi, 60)	(water, 35)
15	(hi, 20)	(use, 9)	(cours, 57)	(mean, 34)
16	(floor, 20)	(mean, 8)	(week, 56)	(gunna, 32)
17	(line, 19)	(golf, 8)	(hous, 56)	(hi, 32)
18	(plant, 19)	(trevor, 8)	(problem, 54)	(price, 32)
19	(famili, 19)	(sprog, 8)	(garden, 52)	(moment, 31)

	NAT
0	(time, 149)
1	(way, 121)
2	(thing, 105)
3	(book, 100)
4	(someth, 92)
5	(lot, 81)
6	(water, 80)
7	(world, 76)
8	(life, 68)
9	(famili, 60)
10	(school, 60)
11	(question, 59)
12	(year, 55)
13	(morn, 55)
14	(countri, 53)
15	(mean, 51)
16	(govern, 51)
17	(day, 51)
18	(gunna, 50)
19	(kind, 50)

Common word analysis: Topical, only nouns kept in addition to more stop words used)

Through using nltk's pos_tag feature, we are able to filter out all non-noun words to achieve some targeted analysis on topics that are discussed throughout the radio transcripti

Word Count Analysis (Nouns):

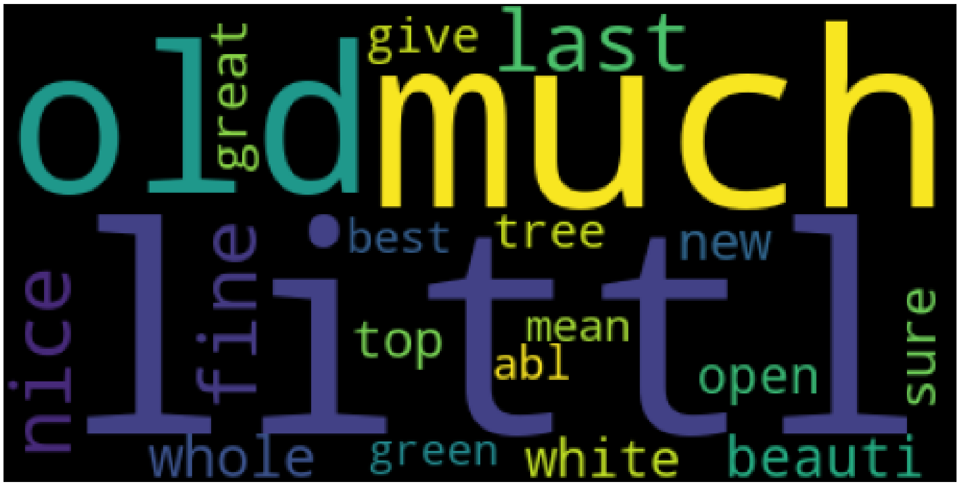
Interesting to note that "time", "year", "morn/morning", "afternoon", "week", "today" and "day" are very commonly used words. These are all "time-related" nouns and this does make sense as generally radios may discuss the daily events of the speakers, resulting in an increased use of time-related nouns to describe experiences and current ongoing events.

Other common nouns include "life", "school" and "famili" (root of "family") are also commonly discussed. Radios generally appeal to an older audience as young adults primarily use the internet to get their entertainment (podcasts on youtube for example). Research shows that the primary age range of radio listeners in Australia (in July 2023) was between 25 and 39, followed by 40-54 and 65+ (See Reference 1). The ages below 24 were substantially less when compared to older age ranges. Generally, after the age of 25 and when heading into the 30s age range, individuals will look towards settling down with family or having kids while being "working" adults, which were common keywords discovered in the analysis.

Following this analysis, the words "plant" and "garden" were also listed as common nouns, but only for the ABCE and COME channels. These channels are specific to Eastern Australia which compass generally Queensland and New South Wales. Interestingly enough, these two states are considered to have the best climate for plants but it's difficult to interpret how much of this is coincidence vs caus adjectives. ons.

[105]:
```
#Comparing Word counts extracting only Adjectives
custom_punctuation = "''""…""...""``" '"' '_'
custom_stopwords = []
word_freqs = {}

for index, row in df_combined.iterrows():
    #JJ: adjective or numeral, ordinal
    #JJR: adjective, comparative
    #JJS: adjective, superlative
    processed = preprocessor.preprocess_keep_pos_tag(row['content'], {"JJ",␣
 ↪"JJR", "JJS"})

    word_freqs[row['prefix']] = get_word_frequencies(processed, top_n=20)

    print(row['prefix'])
    wordcloud = WordCloud(max_words=20).generate(processed)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()

# Combine word frequencies into a DataFrame
df_word_freqs = pd.DataFrame(word_freqs).fillna(0)
```

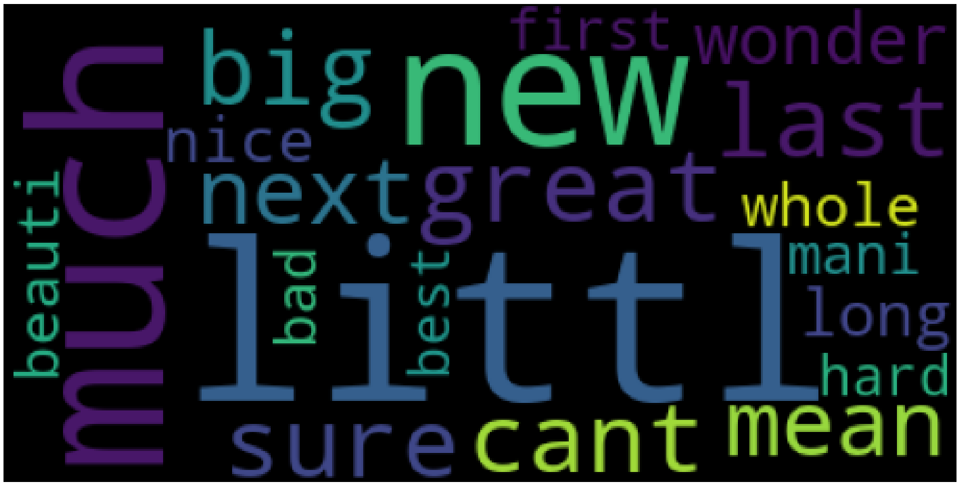
```
ABCE
```


old
much
littl
last
give
great
best
tree
new
nice
fine
top
mean
abl
open
sure
whole
green
white
beauti

ABCNE


littl
thirti
seventeen
sure
particular
certain
nice
big
whole
great
ah
green
new
last
much
australian
old
late
next
alright
thirteen

COME


much
littl
new
big
great
next
last
wonder
first
nice
whole
mani
long
hard
mean
cant
sure
beauti
bad
best

COMNE


littl
much
big
last
mean
mani
sure
fantast
give
old
abl
best
new
real
next
top
first
twenti
great
nice

NAT

[106]:
```
df_word_freqs
```

[106]:

	ABCE	ABCNE	COME	COMNE
0	(littl, 49)	(littl, 23)	(littl, 135)	(much, 79)
1	(old, 39)	(last, 15)	(much, 91)	(littl, 67)
2	(much, 34)	(much, 15)	(new, 72)	(doubl, 57)
3	(last, 24)	(new, 14)	(great, 64)	(last, 47)
4	(nice, 22)	(green, 11)	(big, 61)	(mean, 46)
5	(fine, 21)	(old, 10)	(last, 61)	(big, 44)
6	(whole, 21)	(alright, 10)	(cant, 59)	(mani, 42)
7	(beauti, 19)	(great, 9)	(sure, 58)	(sure, 40)
8	(top, 18)	(thirteen, 9)	(next, 53)	(old, 39)
9	(new, 18)	(seventeen, 8)	(mean, 53)	(great, 38)
10	(sure, 18)	(certain, 7)	(wonder, 51)	(twenti, 36)
11	(great, 18)	(late, 7)	(long, 41)	(new, 36)
12	(white, 18)	(thirti, 7)	(nice, 38)	(fantast, 30)
13	(open, 17)	(next, 7)	(wrong, 34)	(abl, 29)
14	(give, 16)	(particular, 6)	(whole, 34)	(next, 29)
15	(tree, 16)	(australian, 6)	(beauti, 33)	(nice, 27)
16	(mean, 14)	(nice, 6)	(first, 33)	(real, 26)
17	(abl, 14)	(whole, 6)	(mani, 32)	(first, 26)
18	(best, 14)	(big, 6)	(bad, 31)	(top, 25)
19	(green, 13)	(sure, 5)	(hard, 29)	(give, 25)

\

	NAT
0	(littl, 108)
1	(much, 101)
2	(mean, 91)
3	(australian, 82)
4	(big, 64)

```
5         (great, 64)
6          (last, 59)
7           (new, 59)
8        (differ, 56)
9          (sure, 49)
10        (first, 49)
11        (whole, 48)
12        (tripl, 48)
13     (interest, 46)
14         (mani, 45)
15         (cant, 44)
16          (old, 40)
17       (hormon, 38)
18    (australia, 36)
19    (christian, 35)
```

Word Clouds: Adjectives Only

Common adjectives show that “little” is nearly universally the most used adjective. Through examining the transcripts themselves, there are many cases of “little bit” being used and this has caused a large amount of use of the word “little” as a describe adjective.

One key word that appears within the top 20 adjectives used is the word “Christian”. This word is the 20th most used adjective for the National Australia Broadcast and it only appears within the list for NAT, suggesting that national radio could have more discussions that touch serious topics such as religion. It’s still a very limited use of the word Christian, but other radio stations do not have any cases of words relating to religion appearing. Aside from this potential interpretation, there is not much in terms of differences that we can gather from commonly used adjectives.

[108]:
```
#Bigrams

#Comparing Word counts extracting only Adjectives
custom_punctuation = "''""…""...""``" '"' '–'
custom_stopwords = ["wanna"]
bigrams_dict = {}

for index, row in df_combined.iterrows():
    preprocessed = preprocessor.preprocess(row['content'])
    bigrams_list = list(bigrams(preprocessed.split()))
    bigram_counter = Counter(bigrams_list)
    bigrams_dict[row['prefix']] = bigram_counter.most_common(10)

df_bigrams_dict = pd.DataFrame(bigrams_dict).fillna(0)
```

[109]:
```
df_bigrams_dict
```

[109]:
```
                     ABCE                      ABCNE                    COME  \
0      ((year, old), 11)  ((thirteen, hundr), 12)       ((wan, na), 57)
1        ((wan, na), 11)    ((hundr, thirti), 12)    ((morn, morn), 34)
```

```
2    ((root, system), 9)  ((thirti, seventeen), 12)         ((next, year), 27)
3  ((build, consult), 9)   ((seventeen, hundr), 12)            ((bye, bye), 23)
4      ((book, club), 9)       ((hundr, number), 9)           ((year, ago), 23)
5      ((love, bone), 9)          ((time, year), 7)             ((let, go), 21)
6    ((open, garden), 8)         ((nelli, kelli), 7)         ((come, back), 21)
7   ((twenti, minut), 8)          ((green, leav), 7)                ((g, p), 20)
8     ((steel, wool), 8)           ((nake, ladi), 6)         ((dr, graham), 19)
9      ((take, call), 7)           ((last, year), 6)  ((thousand, dollar), 18)

                    COMNE                            NAT
0         ((wan, na), 58)                ((wan, na), 40)
1    ((doubl, doubl), 42)        ((hormon, therapi), 32)
2       ((year, old), 24)           ((hundr, tripl), 30)
3  ((twenti, twenti), 22)                ((let, go), 22)
4     ((youll, find), 20)              ((year, ago), 21)
5     ((year, twelv), 19)         ((children, book), 19)
6   ((doubl, eighti), 18)           ((twenti, year), 18)
7       ((year, ago), 17)               ((tell, us), 18)
8      ((last, year), 15)               ((tripl, j), 18)
9        ((pot, mix), 14)  ((australian, children), 17)
```

[110]:
```python
#Trigrams

#Comparing Word counts extracting only Adjectives
custom_punctuation = "''""…""...""``" '"' '–'
custom_stopwords = []
trigrams_dict = {}

for index, row in df_combined.iterrows():
    preprocessed = preprocessor.preprocess(row['content'])
    trigrams_list = list(trigrams(preprocessed.split()))
    trigram_counter = Counter(trigrams_list)
    trigrams_dict[row['prefix']] = trigram_counter.most_common(10)

df_trigrams_dict = pd.DataFrame(trigrams_dict).fillna(0)
```

[111]:
```python
df_trigrams_dict
```

[111]:
```
                            ABCE                                  ABCNE  \
0    ((fourteen, year, old), 7)    ((thirteen, hundr, thirti), 12)
1  ((nineti, cabbag, tree), 6)   ((hundr, thirti, seventeen), 12)
2     ((cabbag, tree, road), 6)   ((thirti, seventeen, hundr), 12)
3    ((tree, road, bayview), 6)    ((seventeen, hundr, number), 9)
4  ((open, garden, scheme), 5)  ((nelli, kelli, passionfruit), 4)
5    ((twenti, minut, past), 5)                     ((k, e, e), 2)
6  ((sydney, radio, across), 5)            ((pull, someon, leg), 2)
7     ((radio, across, new), 5)             ((club, st, andrew), 2)
```

```
8     ((across, new, south), 5)     ((forecaddi, someon, went), 2)
9       ((new, south, wale), 5)             ((us, tilli, van), 2)

                                 COME                              COMNE  \
0  ((thirteen, thirteen, thirti), 13)  ((doubl, doubl, eighti), 18)
1           ((put, back, switch), 10)   ((doubl, doubl, doubl), 12)
2             ((morn, dr, graham), 9)           ((dot, com, dot), 8)
3    ((sharina, saturday, night), 9)             ((com, dot, u), 8)
4                 ((duh, duh, duh), 8)      ((morn, harvey, morn), 6)
5               ((morn, morn, dr), 8)    ((solvol, citru, soap), 6)
6            ((love, song, dedic), 8)      ((citru, soap, pack), 6)
7                ((dot, com, dot), 7)   ((terracotta, pot, mix), 6)
8                  ((com, dot, u), 7)                ((p, r, dot), 5)
9                ((bye, bye, bye), 6)             ((r, dot, com), 5)

                                     NAT
0                 ((new, south, wale), 16)
1               ((world, refuge, day), 13)
2                   ((give, us, call), 12)
3      ((australian, children, book), 10)
4                ((bye, hundr, tripl), 10)
5             ((australia, talk, back), 7)
6                         ((eh, eh, eh), 7)
7  ((australian, children, literatur), 7)
8          ((hormon, replac, therapi), 7)
9                         ((tt, tt, tt), 7)
```

Engram analysis:

As discussed before, the occurrence of "wanna" shows a use of colloquial language is very common. Bigrams such as "open garden", "green leaves" show that gardening was a discussed topic (as mentioned before) for the ABCE (eastern australian) radio station. Combinations such as "Children book", "book club", "hormonal therapy" also suggest these radio discussions are targeted towards a more mature audience as opposed to the younger generation.

Similarly for Trigrams, There are 13 mentions of "world refugee day" by NAT which supports the previously discussed interpretation that the National Radio Station may cover more serious topics.

Now we group all the Non-commercial radio stations together and the commercial radio stations together and apply similar analysis.

As discussed before we can see that colloquial language ("y'know" and "gunna") make common appearances in non-commercial radio transcripts, likely for the reason to make the radio station more relatable and appealing to a wider range of audiences.

[113]:
```
df_combined['type'] = df_combined['prefix'].apply(lambda x: "COM" if x.
 ↪startswith('COM') else 'non-COM')


#Group by prefix
```

```
df_combined_com = df_combined.groupby('type')['content'].apply(' '.join).
    ↪reset_index()
```

[114]:
```
df_combined_com
```

[114]:

	type	content
0	COM	Good morning and welcome to another Two G B w…
1	non-COM	Thanks for that John Hall now John Hall will …

[115]:
```
#Applying previous investigations to see if there is a different between com␣
    ↪and no-com
custom_punctuation = "''""…""...""``" '"' '-'  # Add any custom punctuation␣
    ↪marks here
custom_stopwords = ["n", "c", "ih", "w"]
preprocessor = TextPreprocessor(custom_punctuation, custom_stopwords)
word_freqs = {}

for index, row in df_combined_com.iterrows():
    processed = preprocessor.preprocess(row['content'])
    word_freqs[row['type']] = get_word_frequencies(processed, top_n=50)

# Combine word frequencies into a DataFrame
df_word_freqs = pd.DataFrame(word_freqs).fillna(0)
```

[116]:
```
df_word_freqs
```

[116]:

	COM	non-COM
0	(uh, 2466)	(uh, 2636)
1	(um, 1159)	(um, 1540)
2	(well, 976)	(think, 706)
3	(yeah, 831)	(yeah, 654)
4	(got, 708)	(well, 650)
5	(that, 700)	(like, 606)
6	(go, 674)	(yknow, 579)
7	(like, 663)	(one, 539)
8	(good, 650)	(that, 502)
9	(get, 638)	(go, 477)
10	(one, 606)	(oh, 472)
11	(think, 593)	(ye, 439)
12	(oh, 578)	(peopl, 416)
13	(im, 501)	(good, 411)
14	(look, 472)	(im, 400)
15	(know, 465)	(got, 397)
16	(your, 460)	(say, 391)
17	(okay, 457)	(get, 388)
18	(say, 428)	(dont, 352)
19	(realli, 427)	(would, 332)

```
20    (dont, 417)   (thank, 325)
21     (two, 404)    (know, 319)
22    (year, 394)   (thing, 298)
23   (thing, 391)    (look, 298)
24   (yknow, 370)   (theyr, 296)
25      (ye, 362)  (realli, 286)
26    (time, 361)    (book, 267)
27   (thank, 334)    (sort, 262)
28     (ive, 328)    (mean, 261)
29   (there, 323)    (year, 250)
30   (would, 315)   (there, 250)
31   (theyr, 299)    (okay, 249)
32   (peopl, 298)    (time, 244)
33    (youv, 284)    (come, 238)
34    (back, 271)     (ive, 234)
35   (three, 267)  (actual, 230)
36    (mean, 263)    (call, 224)
37     (bit, 250)     (two, 224)
38     (see, 248)    (your, 219)
39    (call, 247)   (right, 216)
40   (right, 246)    (much, 216)
41    (sort, 244)     (bit, 211)
42    (want, 237)     (see, 202)
43    (love, 236)   (littl, 194)
44    (morn, 230)    (read, 193)
45    (come, 228)    (take, 187)
46     (lot, 227)    (talk, 182)
47    (much, 224)      (mm, 180)
48   (littl, 223)     (way, 179)
49    (take, 213)    (want, 170)
```

[117]:
```
#Comparing Word counts with more stop words
custom_punctuation = "''""…""...""``" '"' '-'  # Add any custom punctuation␣
 ↪marks here
custom_stopwords = ["n", "c", "ih", "w", "uh", "um", "yes", "yeah","yknow",␣
 ↪"well", "like", "okay", "that", "thats", "one", "got", "oh", "think",␣
 ↪"good", "youre", "dont", "im", "get", "really", "theyre", "know", "ive",␣
 ↪"say", "two", "three", "would", "right", "bit", "word", "thank", "hello",␣
 ↪"youve", "sort", "look", "theres", "mm", "four", "five", "six", "seven",␣
 ↪"eight", "nine", "ten"]
preprocessor = TextPreprocessor(custom_punctuation, custom_stopwords)
word_freqs = {}

for index, row in df_combined_com.iterrows():
    processed = preprocessor.preprocess(row['content'])
    word_freqs[row['type']] = get_word_frequencies(processed, top_n=20)
```

```python
# Combine word frequencies into a DataFrame
df_word_freqs = pd.DataFrame(word_freqs).fillna(0)
```

[118]:
```python
df_word_freqs
```

[118]:

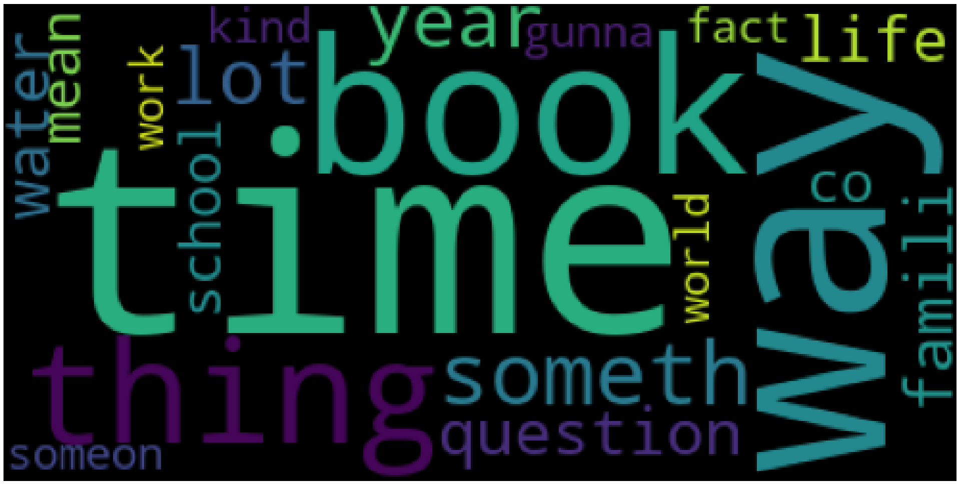
	COM	non-COM
0	(go, 674)	(go, 477)
1	(year, 394)	(peopl, 416)
2	(thing, 391)	(thing, 298)
3	(time, 361)	(book, 267)
4	(peopl, 298)	(mean, 261)
5	(back, 271)	(year, 250)
6	(mean, 263)	(time, 244)
7	(see, 248)	(come, 238)
8	(call, 247)	(actual, 230)
9	(want, 237)	(call, 224)
10	(love, 236)	(much, 216)
11	(morn, 230)	(see, 202)
12	(come, 228)	(littl, 194)
13	(lot, 227)	(read, 193)
14	(much, 224)	(take, 187)
15	(littl, 223)	(talk, 182)
16	(take, 213)	(way, 179)
17	(put, 205)	(want, 170)
18	(need, 202)	(need, 165)
19	(talk, 192)	(us, 162)

[119]:
```python
#Comparing Word counts extracting only nouns
custom_punctuation = "''""…""...""``" '"' '–'
custom_stopwords = ["n", "c", "ih", "w", "uh", "um", "yes", "yeah","yknow",␣
↪"well", "like", "okay", "that", "thats", "one", "got", "oh", "think",␣
↪"good", "youre", "dont", "im", "get", "really", "theyre", "know", "ive",␣
↪"say", "two", "three", "would", "right", "bit", "word", "thank", "hello",␣
↪"youve", "sort", "look", "theres", "mm", "four", "five", "six", "seven",␣
↪"eight", "nine", "ten", "b", "d", "co", "yep", "id"]
preprocessor = TextPreprocessor(custom_punctuation, custom_stopwords)
word_freqs = {}

for index, row in df_combined_com.iterrows():
    processed = preprocessor.preprocess_keep_pos_tag(row['content'], {"NN"})␣
↪#NN = Noun
    word_freqs[row['type']] = get_word_frequencies(processed, top_n=20)

    print(row['type'])
    wordcloud = WordCloud(max_words=20).generate(processed)
    plt.imshow(wordcloud)
    plt.axis("off")
```

```
    plt.show()

# Combine word frequencies into a DataFrame
df_word_freqs = pd.DataFrame(word_freqs).fillna(0)
```

COM

non-COM



[120]:
```
df_word_freqs
```

[120]:

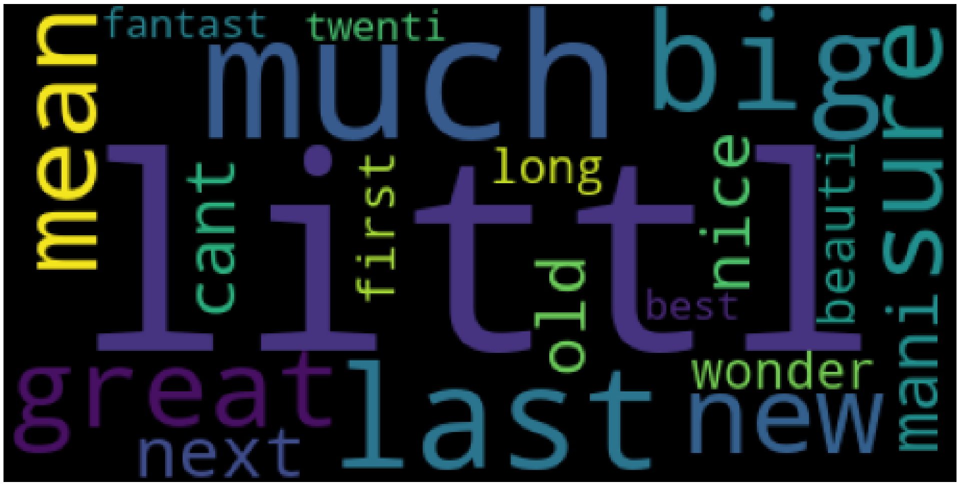
	COM	non-COM
0	(time, 318)	(time, 218)

```
1      (morn, 230)      (book, 165)
2      (year, 230)       (way, 164)
3     (thing, 229)     (thing, 154)
4    (someth, 141)    (someth, 145)
5       (lot, 138)       (lot, 111)
6    (number, 135)      (year, 105)
7       (way, 130)     (water, 103)
8       (day, 118)   (question, 83)
9     (anyth, 115)      (famili, 80)
10    (today, 109)       (world, 79)
11     (life, 106)        (life, 76)
12   (moment, 100)         (day, 75)
13         (hi, 92)       (morn, 70)
14       (week, 91)       (mean, 67)
15      (water, 85)     (school, 67)
16       (work, 81)          (co, 65)
17      (cours, 81)      (gunna, 64)
18       (hous, 78)       (fact, 63)
19    (problem, 77)     (someon, 62)
```

[121]:

```python
#Comparing Word counts extracting only Adjectives
custom_punctuation = "''""…""...""``" '"' '–'
custom_stopwords = []
word_freqs = {}

for index, row in df_combined_com.iterrows():
    #JJ: adjective or numeral, ordinal
    #JJR: adjective, comparative
    #JJS: adjective, superlative
    processed = preprocessor.preprocess_keep_pos_tag(row['content'], {"JJ",␣
 ↪"JJR", "JJS"})

    word_freqs[row['type']] = get_word_frequencies(processed, top_n=20)

    print(row['type'])
    wordcloud = WordCloud(max_words=20).generate(processed)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()

# Combine word frequencies into a DataFrame
df_word_freqs = pd.DataFrame(word_freqs).fillna(0)
```

```
COM
```



non-COM

[122]:
```
df_word_freqs
```

[122]:

	COM	non-COM
0	(littl, 202)	(littl, 180)
1	(much, 170)	(much, 150)
2	(last, 108)	(mean, 106)
3	(new, 108)	(last, 98)
4	(big, 105)	(new, 91)
5	(great, 102)	(great, 91)
6	(mean, 99)	(old, 89)

```
7     (sure, 98)  (australian, 89)
8    (doubl, 86)         (big, 81)
9     (next, 82)       (whole, 75)
10    (mani, 74)        (sure, 72)
11    (cant, 73)      (differ, 70)
12     (old, 67)       (first, 63)
13    (nice, 65)    (interest, 58)
14  (wonder, 60)        (cant, 57)
15   (first, 59)       (tripl, 52)
16  (beauti, 58)        (mani, 51)
17    (long, 54)        (nice, 49)
18  (twenti, 53)       (white, 49)
19    (best, 53)        (long, 47)
```

Commercial vs Non-Commercial

Regarding noun word counts, there is not much of any notable differences or standouts created depending on whether a radio station was commercial vs non-commercial.

Finally the only notable difference between non-commercial radio stations and commercial stations is the word "australian" appeared as the 8th most common noun, likely due to non-commercial radio stations discussing Australia related topics, perhaps world events or current controversies.

In summary, there are some potential differences created by the nature of the radio stations: their audiences, who they are broadcasted to and whether they are non-commercial or commercial. This analysis investigated, discussed and explored some of these cases throughout the dataset and the conclusion is that the audience and location of broadcast can affect the way language is used (colloquial language) and the topics discussed, meanwhile commercial vs non-commercial radio stations did not have any drastic difference between them.

References:

1. Australia: number of radio listeners by age group 2023 | Statista