Themes in most popular songs in South Korean pop in the years 2010-2022

Introduction and research questions

The idea for this project came from looking through similar projects from previous years of the Computational Literacy course, regarding using computational methods in analyzing lyrics of different songs. As I have a personal interest in South Korean pop culture, I came up with a project idea based on lyrical analysis of the most popular Korean pop songs, with the metric of selection being the songs nominated for Song of the Year award at Melon Music Awards, one of the most prestigious awards in South Korean pop culture. While the Song of the Year category exists widely in other award shows, the Melon Music Awards one has the most comprehensible nomination system, as well as is based on Melon Music, which is the biggest streaming website in South Korea, deeming it more representative than most.

All of this taken into consideration, my research questions are the following:
RQ1: What are themes that appear more frequently than others in popular South Korean pop songs based on the nominees and winners of the Song of the Year award in Melon Music Awards?
RQ2: Do different themes appear more frequently in winners than nominees and vice versa? Can this lyrical discrepancy have a bearing on whether a song wins the award?

In other words, I am curious if song lyrics have any bearing on the song’s popularity. My main goal is to analyze the themes of these songs by taking a look at word frequencies in the data and drawing conclusions based on them. I want to practice using various programming frameworks and computational tools and gain a deeper understanding of them for any future project to come.

As for bigger scientific impact, I believe projects like this could help gain an understanding of how much people value lyricism when it comes to listening to music on a regular basis, whether they take it into account at all and if yes, then what kind of lyrics speak to the public. Music is one of the world’s biggest artistic drivers in our times and it would be insightful to dig deeper into one of its aspects. The basis and methods of the project are easily scalable and with more tools and experience at hand, huge amounts of lyrical data could be analyzed in a similar way.

Data and preprocessing

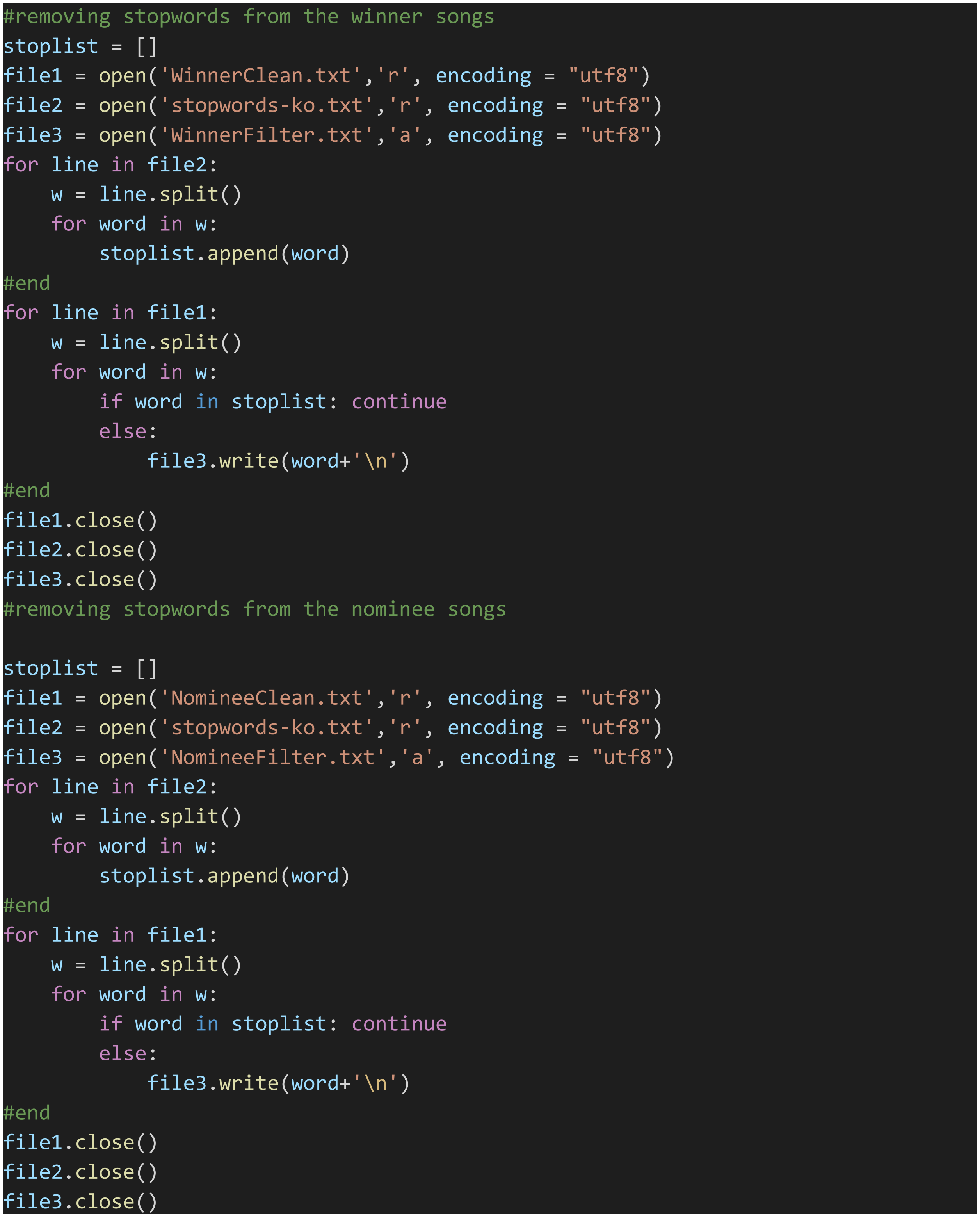
This project is a small data project rather than big data; it consists of 65 Korean songs’ lyrics, acquired free-of-charge from the official Melon Music website. As Melon Music is an official Korean streaming website, all lyrics on it are original and complete. The lyrics are also easily accessible for everyone even without an account on the service, making them easy to acquire. Through looking at old broadcasts of the Melon Music Award show, I can then separate the songs into two categories - **winner songs** and **nominee songs**.

Before acquiring the data and preprocessing it, I want to touch upon the possible biases it may have. Since it is a small scale project, the dataset is not very representative of "popular" songs as a whole, and the choice of data is subjective in itself. The way songs get popular might also come into question, as some of them rise to fame simply due to the popularity of the singer rather than any other factors. Both of these things have to be kept in mind during the process. My decisions about what to include in and exclude from the data are subjective as well, and it has to be kept in mind that someone else might do it differently and get different results. I want to analyze the data based on my own knowledge and draw conclusions from that, but I do not believe it to be the ground truth.

Acquiring the data was quite the tedious task. While there are programming framework projects revolving around scraping lyrics directly from websites, they either grab lyrics only from popular Western-centric websites like Genius (like LyricsGenius) or are a bit too advanced for me to make sense of for now (Lyrics3 code in R studio from this project - coincidentally also taking lyrics from Melon). Therefore, I opted for manually copying and pasting the lyrics into Notepad - one file for winners only (**WinnerSongs.txt**) and one for nominees (**NomineeSongs.txt**). Because my dataset is small, this was possible, but I definitely wouldn't have done it if the data was any bigger; it is not time efficient and it is harder to keep track of what has already been transferred to the file and what has not. I acknowledge that this method of data acquisition is far from ideal and in the future will try to incorporate ways to speed up and revise the process in a more reliable way.

After acquiring the data and separating it into two different files (for winners and nominees), I have to clean it up before I start any visualization or analysis. I chose OpenRefine for the initial clean up, as it is the tool I have worked most with and I find it quite easy to use for basic tasks. I loaded the files into OpenRefine, removed the blank rows, then sorting the file through the text filter so it's alphabetical, which will in turn help me remove duplicate lines (Sort > Text > Reorder rows permanently > Edit cells > Blank down, then you use a blank facet to see which ones are 'true', click those and Remove matching rows). The choice to remove duplicate lines comes from them possibly skewing the results towards whatever is in the repeated lyrics - most of the time these come from choruses, which are made to sound catchy rather than be lyrically profound. These files are saved as **WinnerClean** and **NomineeClean**.

At this point in the preprocessing, I have the text line by line, without blanks and duplicates. Before I try to analyze my data, I want to do one more thing - that is removing stopwords. Stopwords are sets of commonly used words in languages that often do not add much value to text analysis - in English, it would be words such as "how" or "to". Removing them from the text gives me a clearer overview of the important words in the data. To get rid of stopwords, I use the Stopwords Korean collection made by genediazjr on GitHub. I downloaded the text file available on the repository, then used a code someone posted on StackOverflow to run my winner and nominee songs with the stopwords. I saved the output as **WinnerFilter.txt** and **NomineeFilter.txt**.



```
#removing stopwords from the winner songs
stoplist = []
file1 = open('WinnerClean.txt','r', encoding = "utf8")
file2 = open('stopwords-ko.txt','r', encoding = "utf8")
file3 = open('WinnerFilter.txt','a', encoding = "utf8")
for line in file2:
    w = line.split()
    for word in w:
        stoplist.append(word)
#end
for line in file1:
    w = line.split()
    for word in w:
        if word in stoplist: continue
        else:
            file3.write(word+'\n')
#end
file1.close()
file2.close()
file3.close()
#removing stopwords from the nominee songs

stoplist = []
file1 = open('NomineeClean.txt','r', encoding = "utf8")
file2 = open('stopwords-ko.txt','r', encoding = "utf8")
file3 = open('NomineeFilter.txt','a', encoding = "utf8")
for line in file2:
    w = line.split()
    for word in w:
        stoplist.append(word)
#end
for line in file1:
    w = line.split()
    for word in w:
        if word in stoplist: continue
        else:
            file3.write(word+'\n')
#end
file1.close()
file2.close()
file3.close()
```

Extracting word frequencies and making the word cloud

With my data cleaned up, I could finally get to the analysis part and extract specific words and their frequencies. This proved to be trickier than I initially thought and unfortunately, I didn’t achieve everything I wanted. I will touch upon this shortly.

Extracting word frequencies was quite easy thanks to the NLTK package for Python, which is one of the most well-known NLP packages available - and calculating word frequency is one of the simplest and most basic NLP tasks. The code for all the tasks associated with calculating the frequency using NLTK is listed below, with explanations for specific lines of code.

```
import nltk
from nltk import word_tokenize
from nltk.probability import FreqDist
import urllib.request
from matplotlib import pyplot as plt
nltk.download('punkt')
```

```
#loading file in and converting it into a string to be read, then tokenizing
text_file = open('NomineeFilter.txt', 'r', encoding = "utf8")
text = text_file.read().lower()
words = word_tokenize(text)

#calculating the total number of words
print(f"The total number of words in the text is {len(words)}")

#calculating frequency
fdist = FreqDist(words)
fdist.most_common()
```

```
#loading file in and converting it into a string to be read, then tokenizing
text_file = open('WinnerFilter.txt', 'r', encoding = "utf8")
text = text_file.read().lower()
words = word_tokenize(text)

#calculating the total number of words
print(f"The total number of words in the text is {len(words)}")

#calculating frequency
fdist = FreqDist(words)
fdist.most_common()
```

I can then save the output in the form of a list of words with their frequencies in Excel - **WinFreq.xlsx** and **NomFreq.xlsx**. This is where I found myself a bit lost on what to do next. Initially, I wanted to only analyze the nouns - as those are the words one can most commonly use for grasping themes of songs. I wanted to use the Hannanum class from the KoNLPy (Korean NLP) package for Python for POS (part of speech) tagging as I have seen use of it in other similar projects. However, from what I understood, one of the parts of the installation process did not have a proper wheel available for the current version of Python, which rendered the entire process not possible to do.

With this plan failing, I thought I could possibly cluster specific entries in OpenRefine by their meaning. While this is technically possible, it involves a lot of manual filtering of clusters - none of the clustering options are very good at recognizing Korean alphabet nor at matching them based on the nearest neighbour. Manual filtering is an interpretative act that is wrought with bias and based on own knowledge, which would differ from person to person. Therefore, it easily clashes with the idea of easy reproducibility. With all of this taken into consideration, I scrapped this idea as well.

I am sure that there are other ways of doing what I wanted, but at this point, I was left with little time to experiment, so I opted out of POS-tagging and decided to leave the word frequencies as they are - I will simply give short explanations for the most common words in the lists based on the part of speech they represent.

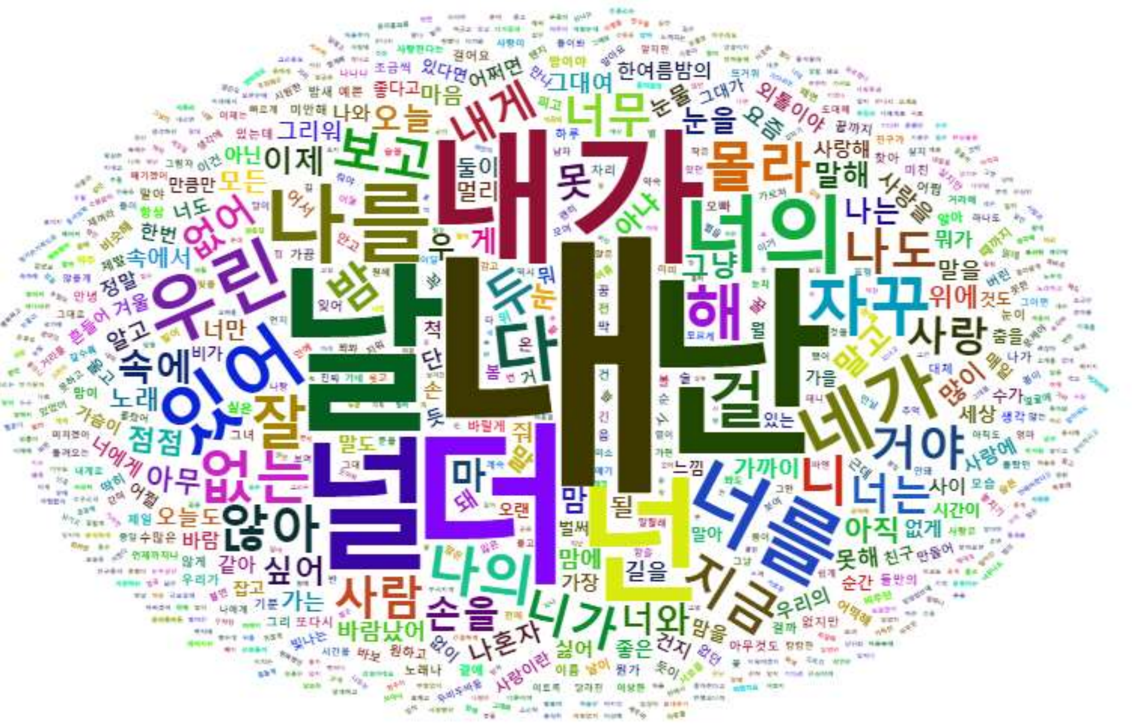
With this out of the way, I can finally generate wordclouds for each of my datasets. Due to the Python Wordcloud package being incompatible with the newest version of the framework and no wheel available so far, I opted for importing my data into R studio and using the Wordcloud2 library. This was easy enough, especially since I have a bit of experience with this particular framework. For generating the wordclouds, I first generated them without any filters, then decided to filter out the words that only appear once, as without clustering them with more abundant words they do not hold significant meaning. The first wordcloud is the nominees' frequency, the second one is the winners'.

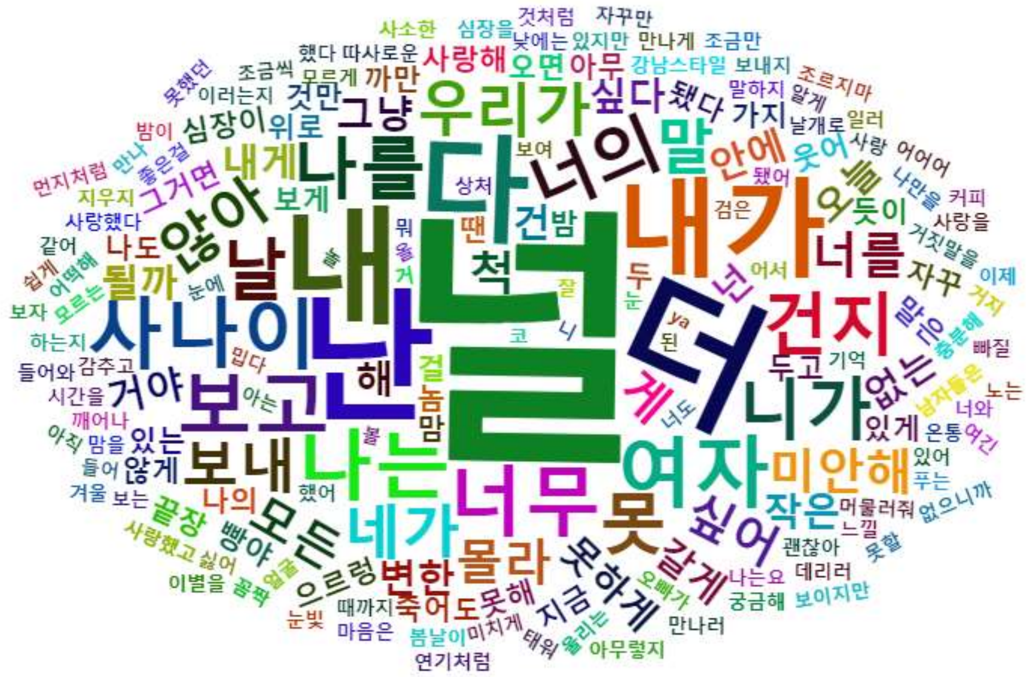
~~~
```{r}
require(devtools)
install_github("lchiffon/wordcloud2")
```

```{r}
library(wordcloud2)
library(tidyverse)
```

```{r}
NomFreq = NomFreq %>% filter(FREQ > 1)
wordcloud2(data = NomFreq)
```

```{r}
WinFreq = WinFreq %>% filter(FREQ > 1)
wordcloud2(data = WinFreq)
```
~~~





I also attempted to load the data into the custom Venn diagram tool made by the Bioinformatics & Evolutionary Genomics of Gent University in Belgium. However, all that is shown on them is the number of words that connect the two sets, rather than the words themselves, and the website does not read the Korean alphabet very well. I did try to romanize the words and create a diagram with a text list based on that, but I don’t believe it’s very readable. The result of the diagram will be included in the analysis below, with a text list attached in the ‘analysis’ folder.

Analysis of results and the project overall

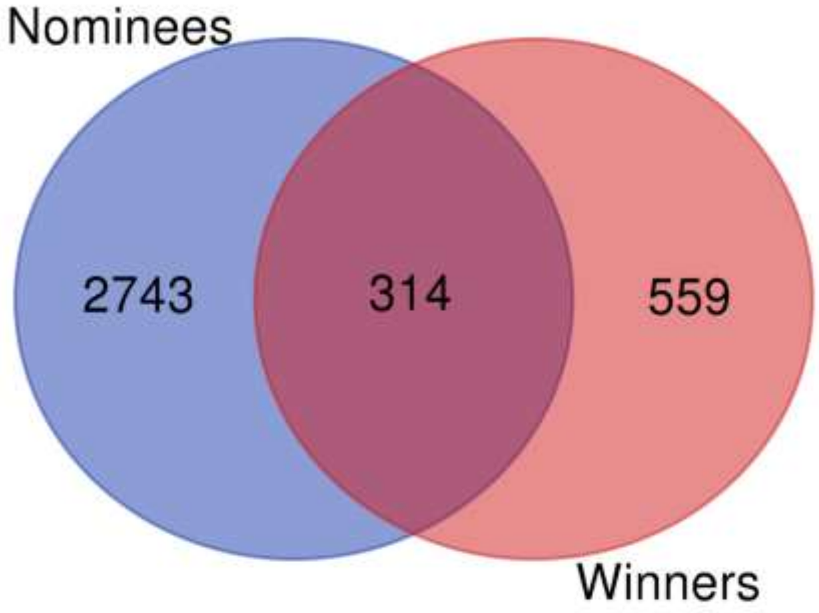
As one can see from the wordclouds, the most frequent words in both sets seem to be pronouns of the first person; 내 (nae), 난 (nan), 날 (nal) all relate to “I, me, my"; the person is talking about themselves and their possessions. This result is not a surprise to me, as pronouns are expected to appear quite frequently even in regular speech, and in songs, they often relate to either the speaker or the person they are singing about. The latter appears as well, right under the first-person pronouns, in the form of 널 (neol) and 넌 (neon) - both possessive pronouns meaning “your”. There is also “us, we, our” in the form of 우리 (uri), 우린 (urin) and such. In fact, all but 1 from the top 10 most frequent words in the nominees set and 6 out of 10 from the winners set are pronouns, which is a significant amount. Pronouns, however, do not give much knowledge to the specific themes of songs, as they appear in many songs with different lyrics and just by themselves do not hold significant thematic value - however, the appearance of pronouns referring to multiple people such as “ours” and “us” could suggest a relationship between them, which gives a clue to the theme.

Before discussing the nouns, which will most likely be the key POS in determining themes, I want to briefly talk about the other common words that appeared at the top of the list. The remaining top 10 word that isn’t a pronoun in the nominees set is the word 더 (deo), which means “more” and is a very common adjective in Korean. It also appears in the top 10 for the winners’. It’s used as a comparative in collocation with other adjectives, which is why it places so high on the frequency list. The remaining word from the winners’ list that isn’t a noun or pronoun is 너무 (neomu) - meaning “a lot”. Its use is parallel to 더 (deo) and neither of them hold a lot of meaning as standalone words.

This brings us to the nouns. It is quite interesting to me that no nouns made the nominees’ list top 10 in terms of frequency, while 2 made it to the top 10 in winners’ list - 여자 (yeoja) and 사나이 (sanai), meaning “woman” and “man” respectively. It is very difficult to assess how much this means in terms of differences between the two sets, as they are small, and the frequency could be chalked up to them appearing in just one song - but it is something interesting to keep in mind. One could hypothesise that appearing more impersonal in lyrics appeals to people, hence the use of these general terms. Further experimenting with this kind of method might reveal some results regarding this. There is a noun of a similar kind that appears quite frequently in the nominees’ set, and that’s 사람 (saram), meaning “person”. Interestingly, this word does not really appear in the winners’ set, and the two words that do appear there that I just discussed, don’t really appear in the nominees’ set. This discrepancy is certainly something to be observed more - I do believe that the word for “person” might be too general of a word, which is why the songs with the gendered words are ‘more popular’ and favored.

When it comes to the word 사랑 (sarang) - “love”, it appears in the nominees’ set more frequently than the winners’. Considering the previous point, it does confuse me a bit - it would make sense to me for the words “woman” and “man” to appear alongside “love”, but it doesn’t happen to be the case. It's hard to dwell over why this seems to have subverted my expectations, perhaps my assumption was simply unfounded.

From other nouns, words associated with "eyes" 눈 (nun), such as 눈물 (nunmul) - "tears" and 눈빛 (nunbit) - "color of one's eyes, expression of one's eyes" appear to be quite popular. Eyes and words associated with them are a good way of expressing feelings textually, especially feelings of happiness and love or comparatively heartbreak and sadness, which would explain the starkly different places in which they are used in lyrics. Similarly, words associated with "hand" 손 (son) are a prominent part of Korean song lyrics, indicating various things like "holding hands" 손을 잡고 (soneul japkko) in a relationship.



I could try to analyze many more words out of these datasets, such as words for time of the day or seasons. However, I would like to wrap my analysis up by pointing an interesting thing out. While the Venn diagram does not show which words are shared and which ones are unique to each set, it does show us an interesting discrepancy between how many of these exist. Coming into this project, I wasn't sure if I would find anything of note or whether I could prove my hypothesis about the themes and lyrics being quite different between winners and nominees of the Song of the Year award. Looking at the numbers on the diagram, however, proves that there might be something there worth looking into further - between the 3616 unique elements of these sets (3057 in the nominees set and 873 in the winners set), they only share 314 of them, a roughly 10% share. That is not as significant as I thought it would be, and just this finding alone makes me curious of what exactly are the discrepancies and how they show.

These results are in no way representative of the whole Korean pop culture, the lyrics of Korean pop songs nor the industry of Korean pop music. The methods and tools I have used are not perfect and they could do with a lot more polishing. Everything I did was subjective of what I think was good for the process and it can be done differently. I tried my best not to introduce unnecessary bias during the process and I have been transparent about my shortcomings. It is only one small project out of many to be done, and I can only hope that it can help with opening a discussion of what kind of lyrics people like in the music they listen to daily and whether they are considered at all.

When it comes to the process, it was more difficult than I thought it would be. I did not expect so many tools to not be working when I started out and I did not think about the amount of manual labor I would have to put into it. But I am very happy that I have managed to conclude this with a result that makes sense and can be put into clear words, with reproducibility kept in mind. It has helped me understand programming frameworks like Python better, I have become more skilled at the basic clean-up of data through OpenRefine, and I also found myself troubleshooting issues more or less efficiently. I managed to answer my research questions and I think I could open the discussion in this field more. Overall, while I am not fully happy with everything, I think this was a very good lesson for me and has made me more confident in working with these sorts of things in the future – with a better overview of the tools and more time on my hands, I believe I can do more than what I have done here.

References

https://bioinformatics.psb.ugent.be/webtools/Venn/

LyricsGenius, https://lyricsgenius.readthedocs.io/en/master/

KOREAN-POP-LYRICS-ANALYSIS-BY-R, https://github.com/feili0820/KOREAN-POP-LYRICS-ANALYSIS-BY-R

Melon, http://melon.com/

NLTK, https://www.nltk.org/

OpenRefine, https://openrefine.org/

https://stackoverflow.com/questions/24593068/removing-stop-words-without-using-nltk-corpus

Stopwords by Genediazjr, https://github.com/stopwords-iso/stopwords-ko

Wordcloud2, https://cran.r-project.org/web/packages/wordcloud2/vignettes/wordcloud.html