Topic Modelling on Pokemon theme songs

In [1]:

```
# import modules
import gensim
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
import random
import re
import seaborn as sns
import spacy as sc
import unicodedata

from nltk.corpus import stopwords
from nltk import WordNetLemmatizer, PorterStemmer
from pprint import pprint
from sklearn.feature_extraction.text import TfidfVectorizer
from gensim import corpora
from gensim.models import CoherenceModel
```

In [2]:

```
# read lyrics
lyrics_df = pd.read_csv('pokemon_lyrics.csv')
lyrics_df.head()
```

Out[2]:

	Season	Lyrics
0	1	I wanna be the very best\nThat no one ever was...
1	2	So You wanna be a master of Pokemon?\nDo you h...
2	3	Pokémon Johto!\n(Too-doo-doo too too-doo)\n(To...
3	4	Pokemon\nPokemon\nPokemon\nLets do it!\nI wann...
4	5	Pokémon!\nNo time to question my moves\nI stic...

In [3]:

```
# 23 theme songs, one per season starting 1998
lyrics_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23 entries, 0 to 22
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Season  23 non-null     int64 
 1   Lyrics  23 non-null     object
dtypes: int64(1), object(1)
memory usage: 496.0+ bytes
```

In [4]:

```
def strip_accents(s):
   return ''.join(c for c in unicodedata.normalize('NFD', s)
                  if unicodedata.category(c) != 'Mn')
```

In [5]:

```
# pre-processing
def preprocess_text(text, selected_tags=None):

    # tokenize
    nlp = sc.load('en_core_web_sm')
    text = nlp(text)
    tokens = [(w.text, w.pos_) for w in text]

    # remove pos tags
    tags = ('PROPN', 'AUX', 'ADP', 'SYM', 'NUM')
    if selected_tags is not None:
        tokens_rm_pos = [token for (token, pos) in tokens if pos in selected_tags]
    else:
        tokens_rm_pos = [token for (token, pos) in tokens]
    # print(tokens_rm_pos)

    # remove punctuation and case normalize
    tokens_rm_pn =  [w.lower() for w in tokens_rm_pos if w.isalpha()]
```

```
    # remove accented characters
    unaccent = [strip_accents(w) for w in tokens_rm_pn]

    # remove stop words
    stop_words = stopwords.words('english')
    stop_words.extend(['na', 'ta', 'to', 'the',
                       'a', 'be', 'pokemon','sinnoh',
                       'johto','advance', 'advanced',
                       'alolan', 'pallet', 'got', 'get', 'yeah']) # extend stop words
    stop_words = set(stop_words)
    tokens_rm_stop = [w for w in unaccent if not w in stop_words]

    # remove vocables
    patterns = [re.compile(r'o+h+'), re.compile(r'd+o+')]
    tokens_rm_voc = [w for w in tokens_rm_stop if not any(pattern.match(w) for pattern in patterns)]

    # lemmatizaton
    wnl = WordNetLemmatizer()
    tokens_lemmed = [wnl.lemmatize(w) for w in tokens_rm_voc]

    # convert to str
    cleaned = ','.join(tokens_lemmed)

    return cleaned
```

In [6]:

```
# create doc-term matrix
def create_dtm(df):
    corpus = [song.split(',') for song in df['Lyrics'].tolist()]
    print(random.sample(corpus,1))
    id2word = corpora.Dictionary(corpus)
    doc_term_matrix = [id2word.doc2bow(doc) for doc in corpus]

    return corpus, id2word, doc_term_matrix
```

In [7]:

```
# running lda
def get_topics(dtm, n_topics, id2word, passes, n_words):
    lda = gensim.models.ldamodel.LdaModel
    ldamodel = lda(corpus=dtm, num_topics=n_topics, id2word=id2word, passes=passes)
    topics = ldamodel.print_topics(num_topics=n_topics, num_words=n_words)
    return ldamodel, topics
```

In [8]:

```
# prep data
nouns_df = lyrics_df.copy(deep=True)
descriptors_df = lyrics_df.copy(deep=True)
lyrics_df['Lyrics'] = lyrics_df['Lyrics'].apply(lambda x: preprocess_text(x))
```

In [9]:

```
# getting specific POS tags
nouns_df['Lyrics'] = nouns_df['Lyrics'].apply(lambda x: preprocess_text(x, ('NOUN')))
descriptors_df['Lyrics'] = descriptors_df['Lyrics'].apply(lambda x: preprocess_text(x, ('ADJ', 'ADV', 'VERB'))
```

In [10]:

```
all_corpus, all_id2word, all_dtm = create_dtm(lyrics_df)
nouns_corpus, nouns_id2word, nouns_dtm = create_dtm(nouns_df)
descriptors_corpus, descriptors_id2word, descriptors_dtm = create_dtm(descriptors_df)
```

```
[['everybody', 'want', 'master', 'everybody', 'want', 'show', 'skill', 'everybody', 'want', 'faster', 'make',
'way', 'top', 'hill', 'time', 'try', 'gon', 'little', 'bit', 'better', 'step', 'climb', 'one', 'step', 'ladder
', 'whole', 'new', 'world', 'live', 'whole', 'new', 'way', 'see', 'whole', 'new', 'place', 'brand', 'new', 'at
titude', 'still', 'catch', 'best', 'everybody', 'want', 'make', 'statement', 'everybody', 'need', 'carve', 'ma
rk', 'stand', 'alone', 'victory', 'circle', 'stake', 'claim', 'music', 'start', 'give', 'best', 'ever', 'take'
, 'best', 'shot', 'learn', 'come', 'together', 'whole', 'new', 'world', 'live', 'whole', 'new', 'way', 'see',
'whole', 'new', 'place', 'brand', 'new', 'attitude', 'still', 'catch', 'best', 'whole', 'new', 'world', 'live'
, 'live', 'live', 'live', 'whole', 'new', 'way', 'see', 'see', 'see', 'see', 'whole', 'new', 'place', 'brand',
'new', 'attitude', 'attitude', 'attitude', 'still', 'catch', 'best', 'whole', 'new', 'world', 'live', 'whole',
'new', 'way', 'see', 'whole', 'new', 'place', 'brand', 'new', 'attitude', 'still', 'catch', 'best', 'ho']]
[['hope', 'dream', 'friend', 'destiny', 'sky', 'courage', 'head']]
[['question', 'stick', 'choose', 'gon', 'right', 'never', 'see', 'run', 'away', 'believe', 'believe', 'win', '
believe', 'want', 'whole', 'see', 'believe', 'gon', 'best', 'cuz', 'go', 'believe']]
```

In [11]:

```
# extract topics
all_topics = get_topics(all_dtm, 5, all_id2word, 50, 7)
```

In [12]:

```
nouns_topics = get_topics(nouns_dtm, 5, nouns_id2word, 50, 7)
descriptors_topics = get_topics(descriptors_dtm, 5, descriptors_id2word, 50, 7)
```

```
topic_list = [all_topics, nouns_topics, descriptors_topics]
for topics in topic_list:
    for t in topics:
        pprint(t)
    print('next list\n')
```

```
<gensim.models.ldamodel.LdaModel object at 0x000001981E0EE640>
[(0,
  '0.029*"always" + 0.028*"come" + 0.027*"together" + 0.026*"way" + '
  '0.023*"one" + 0.021*"hard" + 0.020*"right"'),
 (1,
  '0.053*"catch" + 0.026*"journey" + 0.026*"start" + 0.026*"today" + '
  '0.024*"teach" + 0.023*"world" + 0.019*"u"'),
 (2,
  '0.068*"new" + 0.043*"whole" + 0.029*"best" + 0.026*"live" + 0.026*"see" + '
  '0.022*"attitude" + 0.022*"catch"'),
 (3,
  '0.032*"battle" + 0.032*"never" + 0.032*"unbeatable" + 0.025*"sun" + '
  '0.017*"friend" + 0.017*"day" + 0.017*"undefeatable"'),
 (4,
  '0.038*"hero" + 0.025*"rise" + 0.025*"best" + 0.021*"challenge" + '
  '0.021*"believe" + 0.021*"born" + 0.021*"way"')]
next list

<gensim.models.ldamodel.LdaModel object at 0x000001981E0EE400>
[(0,
  '0.088*"journey" + 0.088*"today" + 0.087*"way" + 0.066*"hero" + '
  '0.037*"challenge" + 0.027*"world" + 0.027*"friend"'),
 (1,
  '0.053*"journey" + 0.052*"way" + 0.029*"path" + 0.029*"heart" + '
  '0.029*"battle" + 0.029*"win" + 0.029*"battling"'),
 (2,
  '0.081*"destiny" + 0.077*"friend" + 0.070*"world" + 0.068*"courage" + '
  '0.061*"dream" + 0.048*"power" + 0.041*"master"'),
 (3,
  '0.054*"sun" + 0.037*"day" + 0.037*"brand" + 0.037*"winner" + 0.037*"week" + '
  '0.037*"world" + 0.020*"battle"'),
 (4,
  '0.096*"way" + 0.070*"attitude" + 0.058*"world" + 0.047*"brand" + '
  '0.047*"place" + 0.036*"winner" + 0.025*"step"')]
next list

<gensim.models.ldamodel.LdaModel object at 0x000001981E0EE1F0>
[(0,
  '0.138*"catch" + 0.072*"new" + 0.067*"best" + 0.054*"whole" + 0.049*"teach" '
  '+ 0.032*"see" + 0.032*"live"'),
 (1,
  '0.070*"born" + 0.043*"ever" + 0.029*"best" + 0.029*"go" + 0.029*"feel" + '
  '0.016*"try" + 0.016*"show"'),
 (2,
  '0.074*"wanna" + 0.046*"believe" + 0.029*"take" + 0.029*"greatest" + '
  '0.029*"live" + 0.029*"see" + 0.028*"gon"'),
 (3,
  '0.052*"together" + 0.052*"rise" + 0.037*"come" + 0.037*"tall" + '
  '0.030*"forever" + 0.026*"know" + 0.023*"right"'),
 (4,
  '0.054*"start" + 0.032*"never" + 0.027*"know" + 0.027*"find" + 0.023*"lose" '
  '+ 0.023*"hard" + 0.023*"win"')]
next list
```

In [13]:

```
# find optimal n_topics

def compute_coherence_values(dictionary, corpus, texts, limit, start=2, step=3):
    """
    Compute c_v coherence for various number of topics

    Parameters:
    ----------
    dictionary : Gensim dictionary
    corpus : Gensim corpus
    texts : List of input texts
    limit : Max num of topics
```

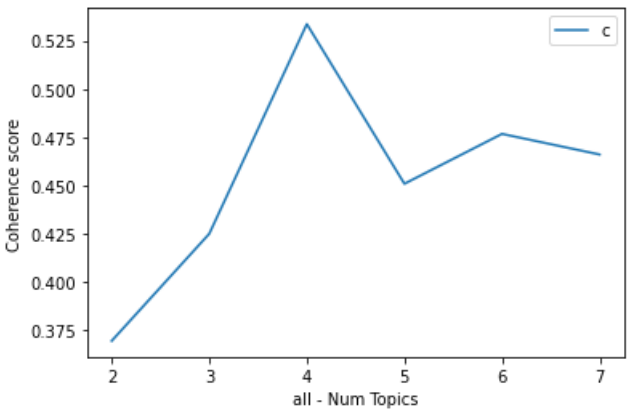
```
    Returns:
    -------
    model_list : List of LDA topic models
    coherence_values : Coherence values corresponding to the LDA model with respective number of topics
    """
    coherence_values = []
    model_list = []
    lda = gensim.models.ldamodel.LdaModel
    for num_topics in range(start, limit, step):
        model = lda(corpus=corpus, num_topics=num_topics, id2word=dictionary, passes=50)
        model_list.append(model)
        coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=dictionary, coherence='c_v')
        coherence_values.append(coherencemodel.get_coherence())

    return model_list, coherence_values
```

In [14]:

```
all_model_list, all_coherence_values = compute_coherence_values(dictionary=all_id2word, corpus=all_dtm, texts=

nouns_model_list, nouns_coherence_values = compute_coherence_values(dictionary=nouns_id2word, corpus=nouns_dtr

descriptors_model_list, descriptors_coherence_values = compute_coherence_values(dictionary=descriptors_id2wor
```

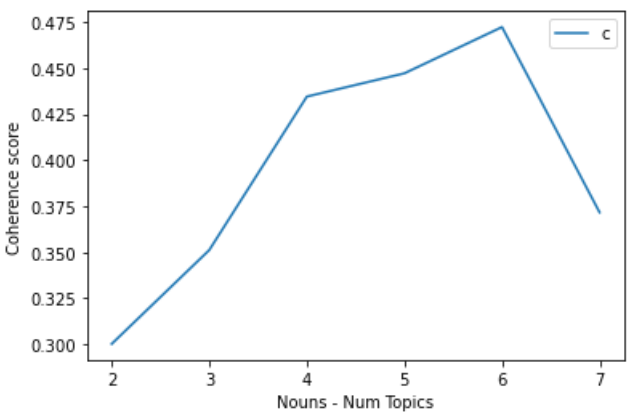
In [15]:

```
# plot coherence values
limit=8; start=2; step=1;
x = range(start, limit, step)
plt.plot(x, all_coherence_values)
plt.xlabel("all - Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```



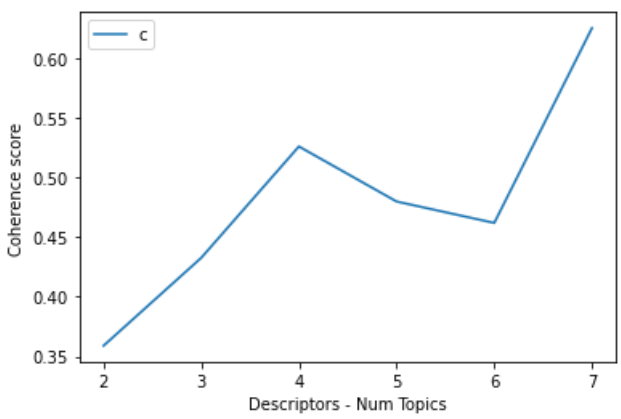
In [16]:

```
# plot coherence values
limit=8; start=2; step=1;
x = range(start, limit, step)
plt.plot(x, nouns_coherence_values)
plt.xlabel("Nouns - Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```



In [17]:

```
# plot coherence values
limit=8; start=2; step=1;
x = range(start, limit, step)
plt.plot(x, descriptors_coherence_values)
plt.xlabel("Descriptors - Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```



In [18]:

```
all_topic_coherence_map = dict(zip(x, all_coherence_values))
for n, cv in all_topic_coherence_map.items():
    print("Num Topics =", n, " has Coherence Value of", round(cv, 4))

all_optimal_n_topics = max(all_topic_coherence_map, key=all_topic_coherence_map.get)
print(f'optimal number of topics: {all_optimal_n_topics}')
```

```
Num Topics = 2  has Coherence Value of 0.3696
Num Topics = 3  has Coherence Value of 0.4251
Num Topics = 4  has Coherence Value of 0.5336
Num Topics = 5  has Coherence Value of 0.4509
Num Topics = 6  has Coherence Value of 0.4767
Num Topics = 7  has Coherence Value of 0.4661
optimal number of topics: 4
```

In [19]:

```
nouns_topic_coherence_map = dict(zip(x, nouns_coherence_values))
for n, cv in nouns_topic_coherence_map.items():
    print("Num Topics =", n, " has Coherence Value of", round(cv, 4))

nouns_optimal_n_topics = max(nouns_topic_coherence_map, key=nouns_topic_coherence_map.get)
print(f'optimal number of topics: {nouns_optimal_n_topics}')
```

```
Num Topics = 2  has Coherence Value of 0.3002
Num Topics = 3  has Coherence Value of 0.3512
Num Topics = 4  has Coherence Value of 0.4346
Num Topics = 5  has Coherence Value of 0.4472
Num Topics = 6  has Coherence Value of 0.4724
Num Topics = 7  has Coherence Value of 0.3716
optimal number of topics: 6
```

In [20]:

```
descriptors_topic_coherence_map = dict(zip(x, descriptors_coherence_values))
for n, cv in descriptors_topic_coherence_map.items():
    print("Num Topics =", n, " has Coherence Value of", round(cv, 4))

descriptors_optimal_n_topics = max(descriptors_topic_coherence_map, key=descriptors_topic_coherence_map.get)
print(f'optimal number of topics: {descriptors_optimal_n_topics}')
```

```
Num Topics = 2  has Coherence Value of 0.3589
Num Topics = 3  has Coherence Value of 0.4325
Num Topics = 4  has Coherence Value of 0.5259
Num Topics = 5  has Coherence Value of 0.4797
Num Topics = 6  has Coherence Value of 0.4618
Num Topics = 7  has Coherence Value of 0.6251
optimal number of topics: 7
```

In [21]:

```
# run final model with optimal n
all_optimal_model, all_optimal_topics = get_topics(all_dtm, n_topics=all_optimal_n_topics, id2word=all_id2wor
for t in all_optimal_topics:
    print(t)
```

```
(0, '0.065*"new" + 0.049*"whole" + 0.029*"see" + 0.029*"live" + 0.025*"best" + 0.025*"attitude" + 0.025*"catch
"')
(1, '0.060*"catch" + 0.032*"best" + 0.026*"teach" + 0.022*"world" + 0.021*"hero" + 0.019*"destiny" + 0.019*"co
urage"')
(2, '0.027*"journey" + 0.027*"start" + 0.027*"today" + 0.025*"way" + 0.020*"together" + 0.018*"battle" + 0.016
*"never"')
(3, '0.030*"master" + 0.026*"wanna" + 0.022*"believe" + 0.018*"hard" + 0.018*"world" + 0.018*"always" + 0.018*
"way"')
```

In [22]:

```
# run final model with optimal n
nouns_optimal_model, nouns_optimal_topics = get_topics(nouns_dtm, n_topics=nouns_optimal_n_topics, id2word=nou
for t in nouns_optimal_topics:
    print(t)
```

```
(0, '0.093*"world" + 0.051*"courage" + 0.051*"way" + 0.050*"friend" + 0.049*"dream" + 0.044*"brand" + 0.044*"d
estiny"')
(1, '0.073*"challenge" + 0.045*"friend" + 0.045*"sun" + 0.045*"heart" + 0.045*"journey" + 0.045*"way" + 0.031*
"day"')
(2, '0.008*"battle" + 0.008*"friend" + 0.008*"destiny" + 0.008*"sky" + 0.008*"power" + 0.008*"path" + 0.008*"t
est"')
(3, '0.090*"winner" + 0.042*"dream" + 0.038*"destiny" + 0.038*"power" + 0.038*"hand" + 0.038*"plan" + 0.038*"m
aster"')
(4, '0.100*"way" + 0.092*"journey" + 0.092*"today" + 0.070*"hero" + 0.039*"friend" + 0.032*"destiny" + 0.032*"
world"')
(5, '0.070*"master" + 0.053*"world" + 0.036*"power" + 0.036*"test" + 0.036*"hand" + 0.036*"skill" + 0.036*"num
ber"')
```

In [23]:

```
# run final model with optimal n
descriptors_optimal_model, descriptors_optimal_topics = get_topics(descriptors_dtm, n_topics=descriptors_optir
for t in descriptors_optimal_topics:
    print(t)
```

```
(0, '0.057*"together" + 0.057*"come" + 0.038*"forever" + 0.036*"always" + 0.029*"right" + 0.029*"tall" + 0.029
*"away"')
(1, '0.067*"wanna" + 0.041*"hold" + 0.041*"live" + 0.041*"greatest" + 0.028*"know" + 0.028*"stand" + 0.028*"ta
ke"')
(2, '0.059*"never" + 0.049*"born" + 0.040*"unbeatable" + 0.040*"ever" + 0.032*"always" + 0.030*"hard" + 0.030*
"best"')
(3, '0.074*"new" + 0.061*"start" + 0.057*"whole" + 0.044*"see" + 0.033*"best" + 0.031*"live" + 0.027*"rise"')
(4, '0.168*"catch" + 0.074*"teach" + 0.051*"best" + 0.039*"know" + 0.034*"pull" + 0.034*"must" + 0.034*"true"'
)
(5, '0.058*"best" + 0.044*"find" + 0.044*"new" + 0.030*"win" + 0.030*"keep" + 0.030*"take" + 0.030*"make"')
(6, '0.059*"go" + 0.032*"feel" + 0.032*"far" + 0.031*"try" + 0.031*"together" + 0.031*"change" + 0.031*"alone"
')
```

In [24]:

```
# find dominant topic

# One of the practical application of topic modeling is to determine what topic a given document is about.

# To find that, we find the topic number that has the highest percentage contribution in that document.

# The format_topics_sentences() function below nicely aggregates this information in a presentable table.

def format_topics_sentences(ldamodel, corpus, texts):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row in enumerate(ldamodel[corpus]):
        row = sorted(row, key=lambda x: (x[1]), reverse=True)

        # Get the Dominant topic, Perc Contribution and Keywords for each document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.Series([int(topic_num), round(prop_topic,4), topic_
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)
```

In [25]:

```
all_topic_docs_keywords = format_topics_sentences(ldamodel=all_optimal_model, corpus=all_dtm, texts=all_corpus

# Format
all_dominant_topic = all_topic_docs_keywords.reset_index()
all_dominant_topic.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', 'Keywords', 'Text']

# Show
all_dominant_topic
```

Out[25]:

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	1.0	0.9864	catch, best, teach, world, hero, destiny, cour...	[wanna, best, one, ever, catch, real, test, tr...
1	1	3.0	0.9865	master, wanna, believe, hard, world, always, w...	[wanna, master, skill, number, wanna, take, ul...
2	2	0.0	0.9940	new, whole, see, live, best, attitude, catch, ...	[everybody, want, master, everybody, want, sho...
3	3	2.0	0.6770	journey, start, today, way, together, battle, ...	[let, wanna, best, like, one, ever, ever, ever...
4	4	3.0	0.9801	master, wanna, believe, hard, world, always, w...	[time, question, move, stick, path, choose, fr...
5	5	1.0	0.9801	catch, best, teach, world, hero, destiny, cour...	[kid, town, brand, new, world, see, know, ahea...
6	6	2.0	0.9808	journey, start, today, way, together, battle, ...	[every, trainer, choice, listen, voice, inside...
7	7	2.0	0.9799	journey, start, today, way, together, battle, ...	[unbeatable, walking, endless, highway, nothin...
8	8	1.0	0.9838	catch, best, teach, world, hero, destiny, cour...	[battle, win, lose, friend, make, road, choose...
9	9	1.0	0.9775	catch, best, teach, world, hero, destiny, cour...	[challenge, brand, new, game, brand, new, worl...
10	10	2.0	0.9625	journey, start, today, way, together, battle, ...	[road, far, home, feel, alone, brave, strong, ...
11	11	2.0	0.9776	journey, start, today, way, together, battle, ...	[sometimes, hard, know, way, supposed, go, dee...
12	12	1.0	0.9606	catch, best, teach, world, hero, destiny, cour...	[hope, dream, friend, work, together, claim, d...
13	13	3.0	0.9735	master, wanna, believe, hard, world, always, w...	[always, hard, journey, begin, hard, find, way...
14	14	2.0	0.9912	journey, start, today, way, together, battle, ...	[new, adventure, another, day, one, challenge,...
15	15	3.0	0.9623	master, wanna, believe, hard, world, always, w...	[next, chapter, ultimate, goal, ready, battle,...
16	16	1.0	0.9925	catch, best, teach, world, hero, destiny, cour...	[verse, wanna, best, like, one, ever, catch, r...
17	17	1.0	0.9704	catch, best, teach, world, hero, destiny, cour...	[kid, quest, best, best, someday, destined, po...
18	18	0.0	0.9482	new, whole, see, live, best, attitude, catch, ...	[stand, tall, know, winner, knock, met, match,...
19	19	0.0	0.9740	new, whole, see, live, best, attitude, catch, ...	[could, used, heat, skin, feel, every, day, li...
20	20	3.0	0.9525	master, wanna, believe, hard, world, always, w...	[preparing, sharing, training, studying, z, bo...
21	21	1.0	0.9723	catch, best, teach, world, hero, destiny, cour...	[rise, prepared, challenge, rise, challenge, r...
22	22	2.0	0.9942	journey, start, today, way, together, battle, ...	[journey, start, today, journey, start, today,...

In [26]:

```
nouns_topic_docs_keywords = format_topics_sentences(ldamodel=nouns_optimal_model, corpus=nouns_dtm, texts=noun

# Format
nouns_dominant_topic = nouns_topic_docs_keywords.reset_index()
nouns_dominant_topic.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', 'Keywords', 'Text']

# Show
nouns_dominant_topic
```

Out[26]:

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	0.0	0.9442	world, courage, way, friend, dream, brand, des...	[one, test, land, power, destiny, friend, worl...
1	1	5.0	0.9701	master, world, power, test, hand, skill, numbe...	[wanna, master, skill, number, step, courage, ...
2	2	0.0	0.9786	world, courage, way, friend, dream, brand, des...	[skill, way, top, hill, time, bit, step, step,...
3	3	3.0	0.9302	winner, dream, destiny, power, hand, plan, mas...	[one, test, cause, life, time, skils, world, w...
4	4	0.0	0.6268	world, courage, way, friend, dream, brand, des...	[time, move, path, friend, fight, dream, chanc...
5	5	4.0	0.9401	way, journey, today, hero, friend, destiny, wo...	[kid, brand, world, battle, step, way, hero, w...
6	6	0.0	0.9301	world, courage, way, friend, dream, brand, des...	[choice, voice, battle, winner, dream, dream, ...
7	7	5.0	0.9164	master, world, power, test, hand, skill, numbe...	[highway, friend, test, earth, land, sea, sky,...
8	8	3.0	0.9581	winner, dream, destiny, power, hand, plan, mas...	[battle, friend, road, stuff, mind, courage, d...
9	9	0.0	0.9403	world, courage, way, friend, dream, brand, des...	[challenge, brand, game, brand, world, rival, ...
10	10	4.0	0.8953	way, journey, today, hero, friend, destiny, wo...	[road, home, destiny, hero, world, friend, hero]
11	11	1.0	0.8803	challenge, friend, sun, heart, journey, way, d...	[way, heart, friend, life, quest, battle]
12	12	0.0	0.8956	world, courage, way, friend, dream, brand, des...	[hope, dream, friend, destiny, sky, courage, h...
13	13	1.0	0.9067	challenge, friend, sun, heart, journey, way, d...	[journey, way, friend, power, heart, win, path...
14	14	4.0	0.9701	way, journey, today, hero, friend, destiny, wo...	[adventure, day, challenge, way, sense, side, ...
15	15	5.0	0.8323	master, world, power, test, hand, skill, numbe...	[chapter, goal, battle, way]
16	16	0.0	0.9730	world, courage, way, friend, dream, brand, des...	[one, test, cause, land, power, destiny, frien...
17	17	4.0	0.9507	way, journey, today, hero, friend, destiny, wo...	[kid, quest, power, glory, test, way, story, h...
18	18	3.0	0.8333	winner, dream, destiny, power, hand, plan, mas...	[winner, match, beginner, winner]
19	19	1.0	0.9583	challenge, friend, sun, heart, journey, way, d...	[heat, skin, day, bit, day, fun, sun, sun, wee...
20	20	1.0	0.8805	challenge, friend, sun, heart, journey, way, d...	[training, z, bonding, battling, destiny, moon]
21	21	1.0	0.9357	challenge, friend, sun, heart, journey, way, d...	[challenge, challenge, champion, heart, challe...
22	22	4.0	0.9833	way, journey, today, hero, friend, destiny, wo...	[journey, today, journey, today, world, way, a...

In [27]:

```
descriptors_topic_docs_keywords = format_topics_sentences(ldamodel=descriptors_optimal_model, corpus=descript

# Format
descriptors_dominant_topic = descriptors_topic_docs_keywords.reset_index()
descriptors_dominant_topic.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', 'Keywords', 'Text

# Show
descriptors_dominant_topic
```

Out[27]:

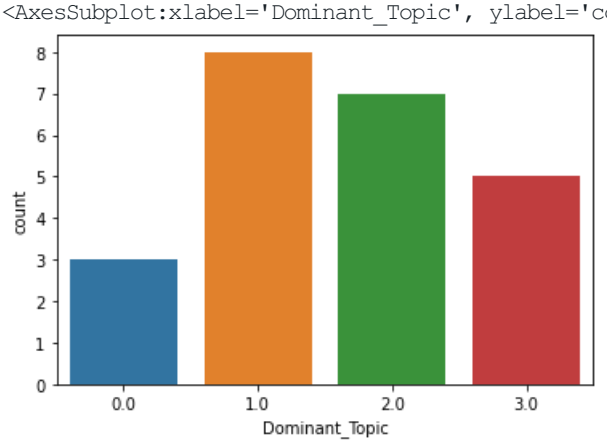
	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	4.0	0.9774	catch, teach, best, know, pull, must, true, de...	[wanna, best, ever, catch, real, train, travel...
1	1	1.0	0.9670	wanna, hold, live, greatest, know, stand, take...	[wanna, take, ultimate, find, bold, risk, forg...
2	2	3.0	0.9895	new, start, whole, see, best, live, rise, beli...	[want, want, show, want, faster, make, try, go...
3	3	2.0	0.9609	never, born, unbeatable, ever, always, hard, b...	[wanna, best, ever, ever, ever, catch, real, t...
4	4	3.0	0.9626	new, start, whole, see, best, live, rise, beli...	[question, stick, choose, gon, right, never, s...
5	5	3.0	0.5227	new, start, whole, see, best, live, rise, beli...	[new, see, know, ahead, wo, best, much, learn,...
6	6	0.0	0.6896	together, come, forever, always, right, tall, ...	[listen, inside, know, may, long, may, come, g...
7	7	2.0	0.9627	never, born, unbeatable, ever, always, hard, b...	[unbeatable, walking, endless, never, give, ne...
8	8	5.0	0.9643	best, find, new, win, keep, take, make, strong...	[win, lose, make, choose, right, make, find, s...
9	9	5.0	0.9428	best, find, new, win, keep, take, make, strong...	[new, new, new, fight, stop, play, smart, move...
10	10	6.0	0.9220	go, feel, far, try, together, change, alone, s...	[far, feel, alone, strong, together, change, t...
11	11	1.0	0.9682	wanna, hold, live, greatest, know, stand, take...	[sometimes, hard, know, supposed, go, deep, in...
12	12	2.0	0.9219	never, born, unbeatable, ever, always, hard, b...	[work, together, claim, reaching, willing, try...
13	13	2.0	0.9570	never, born, unbeatable, ever, always, hard, b...	[always, hard, begin, hard, find, hard, make, ...
14	14	0.0	0.9828	together, come, forever, always, right, tall, ...	[new, come, know, simple, feel, right, make, l...
15	15	0.0	0.9495	together, come, forever, always, right, tall, ...	[next, ultimate, ready, brave, bold, know, gon...
16	16	4.0	0.9862	catch, teach, best, know, pull, must, true, de...	[wanna, best, ever, catch, real, train, travel...
17	17	5.0	0.9046	best, find, new, win, keep, take, make, strong...	[best, best, someday, destined, know, tell, fa...
18	18	3.0	0.9219	new, start, whole, see, best, live, rise, beli...	[stand, tall, know, knock, met, catch, stand, ...
19	19	4.0	0.8927	catch, teach, best, know, pull, must, true, de...	[could, used, feel, little, stronger, wish, lo...
20	20	4.0	0.9047	catch, teach, best, know, pull, must, true, de...	[preparing, sharing, studying, laughing, crazy...
21	21	3.0	0.9387	new, start, whole, see, best, live, rise, beli...	[rise, prepared, rise, rise, rise, take, train...
22	22	3.0	0.9872	new, start, whole, see, best, live, rise, beli...	[start, start, big, big, know, find, together,...

In [28]:

```
# distribution of topics

sns.countplot(x='Dominant_Topic', data=all_dominant_topic)
```

Out[28]:

```
<AxesSubplot:xlabel='Dominant_Topic', ylabel='count'>
```

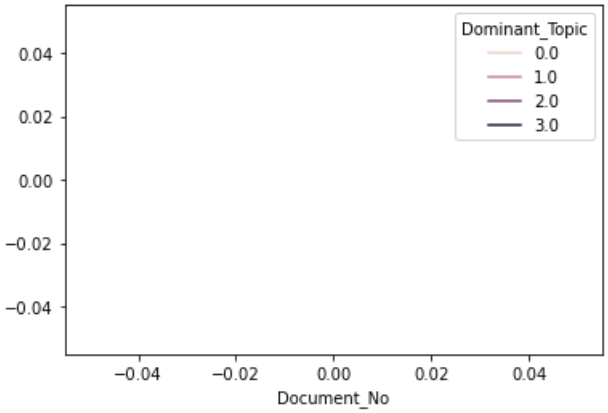


In [29]:

```
# distribution of topics over time
sns.lineplot(x='Document_No', hue='Dominant_Topic', data=all_dominant_topic)
```

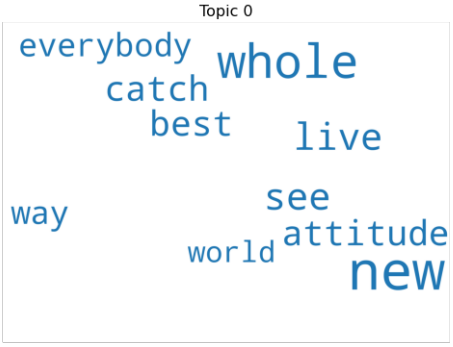
Out[29]:

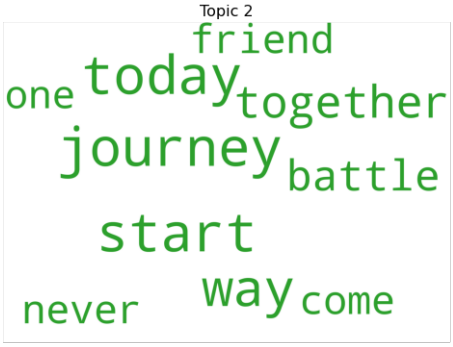
```
<AxesSubplot:xlabel='Document_No'>
```

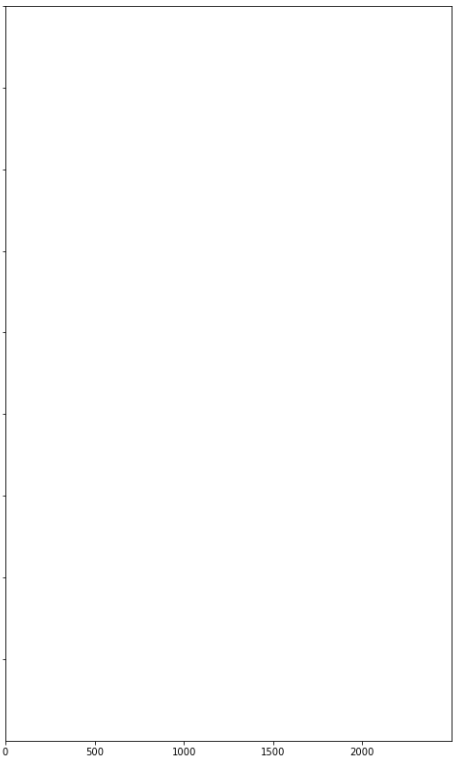


In [30]:

```
# wordcloud of topics
# 1. Wordcloud of Top N words in each topic
from wordcloud import WordCloud, STOPWORDS
import matplotlib.colors as mcolors

cols = [color for name, color in mcolors.TABLEAU_COLORS.items()]  # more colors: 'mcolors.XKCD_COLORS'

cloud = WordCloud(background_color='white',
                  width=2500,
                  height=1800,
                  max_words=10,
                  colormap='tab10',
                  color_func=lambda *args, **kwargs: cols[i],
                  prefer_horizontal=1.0)

topics = all_optimal_model.show_topics(formatted=False)

fig, axes = plt.subplots(2, 3, figsize=(20,20), sharex=True, sharey=True)

for i, ax in enumerate(axes.flatten()):
    try:
        fig.add_subplot(ax)
        topic_words = dict(topics[i][1])
        cloud.generate_from_frequencies(topic_words, max_font_size=300)
        plt.gca().imshow(cloud)
        plt.gca().set_title('Topic ' + str(i), fontdict=dict(size=16))
        plt.gca().axis('off')
    except:
        pass

plt.subplots_adjust(wspace=0, hspace=0)
plt.axis('off')
plt.margins(x=0, y=0)
plt.tight_layout()
plt.show()
```







In [31]:

```
import pyLDAvis
import pyLDAvis.gensim_models
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim_models.prepare(all_optimal_model, all_dtm, dictionary=all_optimal_model.id2word)
vis
```

Out[31]:

Selected Topic: 0 Previous Topic Next Topic Clear Topic

Slide to adjust relevance metric:(2)
λ = 1