In [1]:
```
import numpy as np
import pandas as pd
import gensim
import nltk
import re
import spacy
from gensim.utils import simple_preprocess
import gensim.corpora as corpora
from gensim.models import CoherenceModel
# Plotting tools
import pyLDAvis
import pyLDAvis.gensim  # don't skip this
import matplotlib.pyplot as plt
%matplotlib inline


```

In [32]:
```
!pip install pyLDAvis
```

```
Collecting pyLDAvis
  Downloading pyLDAvis-3.1.0.tar.gz (1.7 MB)
Requirement already satisfied: wheel>=0.23.0 in c:\users\shafeerenbd\anaconda3
Requirement already satisfied: numpy>=1.9.2 in c:\users\shafeerenbd\anaconda3\
Requirement already satisfied: scipy>=0.18.0 in c:\users\shafeerenbd\anaconda3
Requirement already satisfied: joblib>=0.8.4 in c:\users\shafeerenbd\anaconda3
Requirement already satisfied: jinja2>=2.7.2 in c:\users\shafeerenbd\anaconda3
Requirement already satisfied: numexpr in c:\users\shafeerenbd\anaconda3\lib\s
Requirement already satisfied: future in c:\users\shafeerenbd\anaconda3\lib\si
Collecting funcy
  Downloading funcy-1.15-py2.py3-none-any.whl (32 kB)
Requirement already satisfied: pandas>=0.17.0 in c:\users\shafeerenbd\anaconda
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\shafeerenbd\anacor
Requirement already satisfied: pytz>=2017.2 in c:\users\shafeerenbd\anaconda3\
Requirement already satisfied: python-dateutil>=2.6.1 in c:\users\shafeerenbd\
Requirement already satisfied: six>=1.5 in c:\users\shafeerenbd\anaconda3\lib\
Building wheels for collected packages: pyLDAvis
  Building wheel for pyLDAvis (setup.py): started
  Building wheel for pyLDAvis (setup.py): finished with status 'done'
  Created wheel for pyLDAvis: filename=pyLDAvis-3.1.0-py2.py3-none-any.whl siz
  Stored in directory: c:\users\shafeerenbd\appdata\local\pip\cache\wheels\74\
Successfully built pyLDAvis
Installing collected packages: funcy, pyLDAvis
Successfully installed funcy-1.15 pyLDAvis-3.1.0
```

In [2]:
```
dt=pd.read_csv('K8 Reviews v0.2.csv')
```

In [4]:
```
1 dt.head(10)
```

Out[4]:

	sentiment	review
0	1	Good but need updates and improvements
1	0	Worst mobile i have bought ever, Battery is dr...
2	1	when I will get my 10% cash back.... its alrea...
3	1	Good
4	0	The worst phone everThey have changed the last...
5	0	Only I'm telling don't buyI'm totally disappoi...
6	1	Phone is awesome. But while charging, it heats...
7	0	The battery level has worn down
8	0	It's over hitting problems...and phone hanging...
9	0	A lot of glitches dont buy this thing better g...

In [6]:
```
1 dt.shape
```

Out[6]: (14675, 2)

In [5]:
```
1 data=dt['review'].values.tolist()
```

In [6]:
```
1 data
```

Out[6]:
```
['Good but need updates and improvements',
 "Worst mobile i have bought ever, Battery is draining like hell, backup is or
s biggest lie from Amazon & Lenove which is not at all expected, they are maki
5 hours to be fully charged.Don't know how Lenovo will survive by making full
 'when I will get my 10% cash back.... its already 15 January..',
 'Good',
 'The worst phone everThey have changed the last phone but the problem is stil
 "Only I'm telling don't buyI'm totally disappointedPoor batteryPoor cameraWas
 'Phone is awesome. But while charging, it heats up allot..Really a genuine re
 'The battery level has worn down',
 "It's over hitting problems...and phone hanging problems Lenovo k 8 note...sc
by lenovo",
 'A lot of glitches dont buy this thing better go for some other options',
 'Wrost',
 'Good phone but charger not working / damage within 2 months.',
 "Don't purchase this item, It is so much of heating &Battery life is very poc
 'I have faced the battery problem and motherboard problem with in 8 months. T
 'Very good phone slim good battry backup good screen love it',
```

In [7]:
```
data=[re.sub('\s+',' ',sent) for sent in data]

data=[re.sub("\'","",sent) for sent in data]
#data = data.lower()
```

```
<>:1: DeprecationWarning: invalid escape sequence \s
<>:1: DeprecationWarning: invalid escape sequence \s
<>:1: DeprecationWarning: invalid escape sequence \s
<ipython-input-7-82129bff50da>:1: DeprecationWarning: invalid escape sequence
  data=[re.sub('\s+',' ',sent) for sent in data]
```

In [8]:
```
data
```

Out[8]:
```
['Good but need updates and improvements',
 'Worst mobile i have bought ever, Battery is draining like hell, backup is or
s biggest lie from Amazon & Lenove which is not at all expected, they are maki
5 hours to be fully charged.Dont know how Lenovo will survive by making full c
 'when I will get my 10% cash back.... its already 15 January..',
 'Good',
 'The worst phone everThey have changed the last phone but the problem is stil
 'Only Im telling dont buyIm totally disappointedPoor batteryPoor cameraWaste
 'Phone is awesome. But while charging, it heats up allot..Really a genuine re
 'The battery level has worn down',
 'Its over hitting problems...and phone hanging problems Lenovo k 8 note...so
lenovo',
 'A lot of glitches dont buy this thing better go for some other options',
 'Wrost',
 'Good phone but charger not working / damage within 2 months.',
 'Dont purchase this item, It is so much of heating &Battery life is very poor
 'I have faced the battery problem and motherboard problem with in 8 months. T
 'Very good phone slim good battry backup good screen love it',
```

In [9]:
```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))

data_words = list(sent_to_words(data))

print(data_words[:2])
```

```
[['good', 'but', 'need', 'updates', 'and', 'improvements'], ['worst', 'mobile'
nly', 'to', 'hours', 'with', 'internet', 'uses', 'even', 'if', 'put', 'mobile'
'lenove', 'which', 'is', 'not', 'at', 'all', 'expected', 'they', 'are', 'makir
ke', 'it', 'takes', 'at', 'least', 'to', 'hours', 'to', 'be', 'fully', 'charge
'please', 'don', 'go', 'for', 'this', 'else', 'you', 'will', 'regret', 'like',
```

In [10]:
```
data_words
```

Out[10]:
```
[['good', 'but', 'need', 'updates', 'and', 'improvements'],
 ['worst',
  'mobile',
  'have',
  'bought',
  'ever',
  'battery',
  'is',
  'draining',
  'like',
  'hell',
  'backup',
  'is',
  'only',
  'to',
  'hours',
  'with',
  'internet',
```

In [11]:
```
# Build the bigram and trigram models
# min_count (float, optional) - Ignore all words and bigrams with total c


bigram = gensim.models.Phrases(data_words, min_count=5, threshold=100) # 
trigram = gensim.models.Phrases(bigram[data_words], threshold=100)

# Faster way to get a sentence clubbed as a trigram/bigram
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

# See trigram example
print(trigram_mod[bigram_mod[data_words[0]]])
```

```
['good', 'but', 'need', 'updates', 'and', 'improvements']
```

In [12]:
```
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['from', 'subject', 're', 'edu', 'use'])
```

In [13]:
```
# Define functions for stopwords, bigrams, trigrams and lemmatization
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not in 

def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

def make_trigrams(texts):
    return [trigram_mod[bigram_mod[doc]] for doc in texts]

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    """https://spacy.io/api/annotation"""
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in 
    return texts_out
```

In [14]:
```
nlp=spacy.load('en_core_web_sm')
```

In [15]:
```
data_words_nostopwords=remove_stopwords(data_words)
```

In [16]:
```
data_words_nostopwords
```

Out[16]:
```
[['good', 'need', 'updates', 'improvements'],
 ['worst',
  'mobile',
  'bought',
  'ever',
  'battery',
  'draining',
  'like',
  'hell',
  'backup',
  'hours',
  'internet',
  'uses',
  'even',
  'put',
  'mobile',
  'idle',
  'getting',
```

In [17]:
```
data_words_bigrams=make_bigrams(data_words_nostopwords)
```

In [18]:
```
1 data_words_bigrams
```

Out[18]:
```
[['good', 'need', 'updates', 'improvements'],
 ['worst',
  'mobile',
  'bought',
  'ever',
  'battery',
  'draining',
  'like',
  'hell',
  'backup',
  'hours',
  'internet',
  'uses',
  'even',
  'put',
  'mobile',
  'idle',
  'getting',
```

In [19]:
```
1 nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
2 data_lemmatized = lemmatization(data_words_bigrams, allowed_postags=['NOU
```

In [20]:
```
1 data_lemmatized
```

Out[20]:
```
[['good', 'need', 'update', 'improvement'],
 ['bad',
  'buy',
  'ever',
  'battery',
  'drain',
  'hour',
  'internet',
  'use',
  'even',
  'put',
  'mobile',
  'idle',
  'get',
  'discharge',
  'big',
  'lie',
  'expect',
```

In [21]:
```
# Create Dictionary
id2word = corpora.Dictionary(data_lemmatized)

# Create Corpus
texts = data_lemmatized

# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]

# View
print(corpus[:1])
```

```
[[(0, 1), (1, 1), (2, 1), (3, 1)]]
```

In [22]:
```
# Human readable format of corpus (term-frequency)
[[(id2word[id], freq) for id, freq in cp] for cp in corpus[:1]]
```

Out[22]:
```
[[('good', 1), ('improvement', 1), ('need', 1), ('update', 1)]]
```

In [23]:
```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=20,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)
```

In [24]:
```
1 lda_model.print_topics()
```

Out[24]:
```
[(0,
  '0.388*"feature" + 0.275*"great" + 0.197*"go" + 0.000*"compare" + 0.000*"sup
h"'),
 (1,
  '0.376*"battery" + 0.131*"heat" + 0.108*"awesome" + 0.101*"drain" + 0.100*"m
 (2,
  '0.151*"network" + 0.110*"backup" + 0.105*"full" + 0.072*"speed" + 0.070*"lo
 (3,
  '0.395*"product" + 0.303*"work" + 0.161*"well" + 0.079*"excellent" + 0.000*"
*"waste"'),
 (4,
  '0.478*"camera" + 0.182*"nice" + 0.182*"quality" + 0.033*"amazing" + 0.023*"
000*"front"'),
 (5,
  '0.277*"buy" + 0.253*"issue" + 0.181*"heating" + 0.105*"money" + 0.048*"deli
 (6,
  '0.353*"problem" + 0.093*"hang" + 0.059*"service" + 0.047*"thank" + 0.035*"c
k"'),
 (7,
  '0.197*"screen" + 0.192*"even" + 0.092*"many" + 0.079*"option" + 0.069*"want
l"'),
 (8,
  '0.119*"display" + 0.108*"processor" + 0.074*"speaker" + 0.063*"say" + 0.060
e"'),
 (9,
  '0.250*"mobile" + 0.196*"note" + 0.170*"fast" + 0.152*"use" + 0.057*"can" +
 (10,
  '0.269*"charge" + 0.163*"poor" + 0.089*"dual" + 0.087*"month" + 0.063*"turbo
 (11,
  '0.276*"also" + 0.162*"update" + 0.141*"back" + 0.105*"app" + 0.050*"new" +
 (12,
  '0.834*"phone" + 0.049*"return" + 0.033*"first" + 0.023*"still" + 0.009*"ama
t"'),
 (13,
  '0.218*"performance" + 0.103*"sound" + 0.093*"look" + 0.070*"average" + 0.05
c"'),
 (14,
  '0.290*"time" + 0.231*"take" + 0.137*"hour" + 0.075*"lot" + 0.038*"suggest"
 (15,
  '0.812*"good" + 0.119*"price" + 0.010*"point" + 0.008*"already" + 0.007*"imp
*"front"'),
 (16,
  '0.215*"day" + 0.109*"device" + 0.073*"usage" + 0.071*"need" + 0.069*"find"
 (17,
  '0.415*"bad" + 0.122*"call" + 0.110*"low" + 0.072*"purchase" + 0.053*"ever"
 (18,
  '0.189*"range" + 0.188*"really" + 0.149*"worth" + 0.147*"show" + 0.070*"alwa
e"'),
 (19,
  '0.448*"get" + 0.086*"make" + 0.078*"expect" + 0.067*"be" + 0.042*"internet"
```

In [25]:
```
1 doc_lda = lda_model[corpus]
```

In [26]:
```
1 doc_lda
```

Out[26]:
```
<gensim.interfaces.TransformedCorpus at 0x2d327be9708>
```

In [27]:
```
1 print('Perplexity: ',lda_model.log_perplexity(corpus))
```

```
Perplexity:  -12.9200885952944
```

In [28]:
```
1 # Compute Coherence Score
2 coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lemmatiz
3 coherence_lda = coherence_model_lda.get_coherence()
4 print('\nCoherence Score: ', coherence_lda)
```

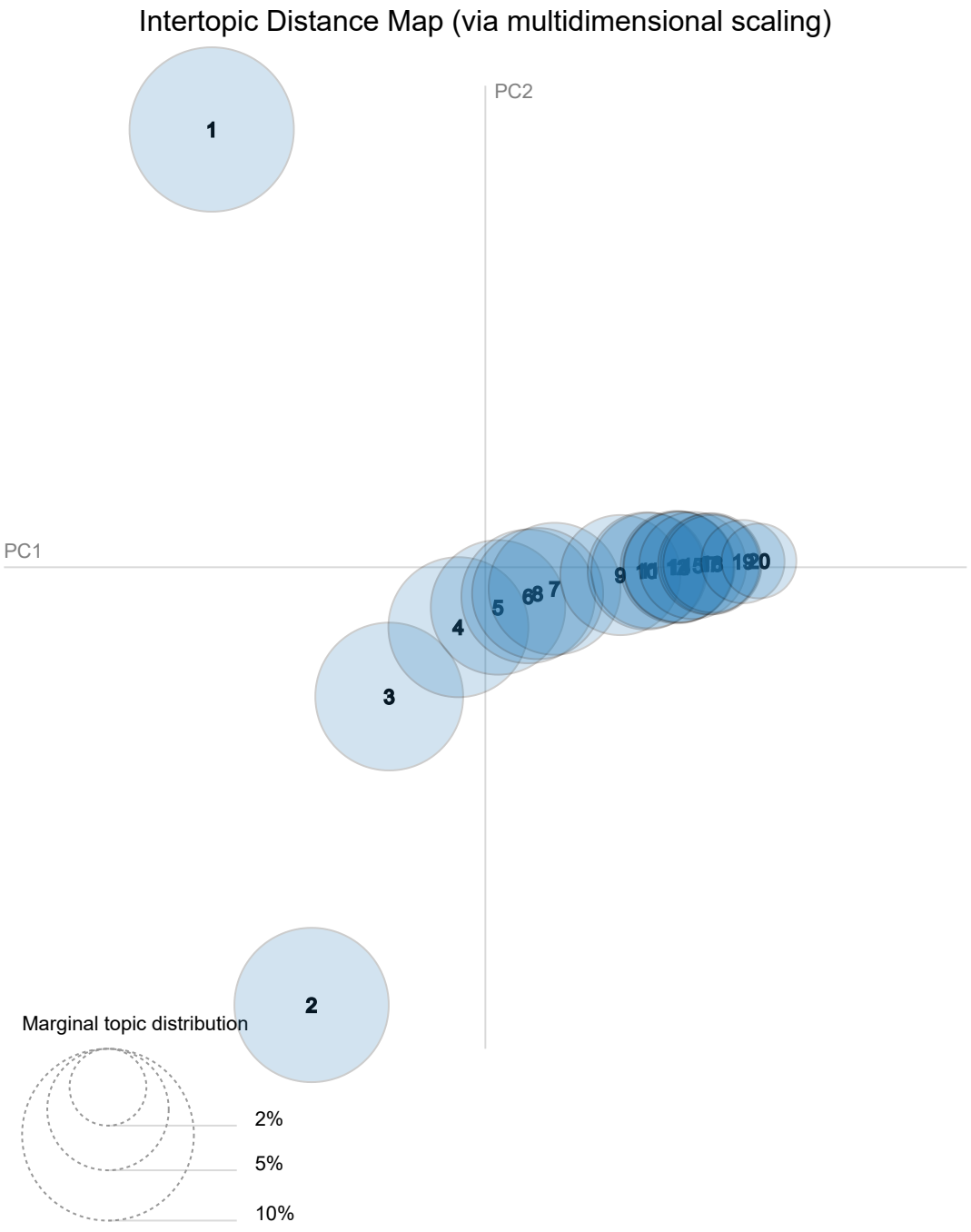
```
Coherence Score:  0.41702860076164194
```

In [29]:

```
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```

Out[29]:

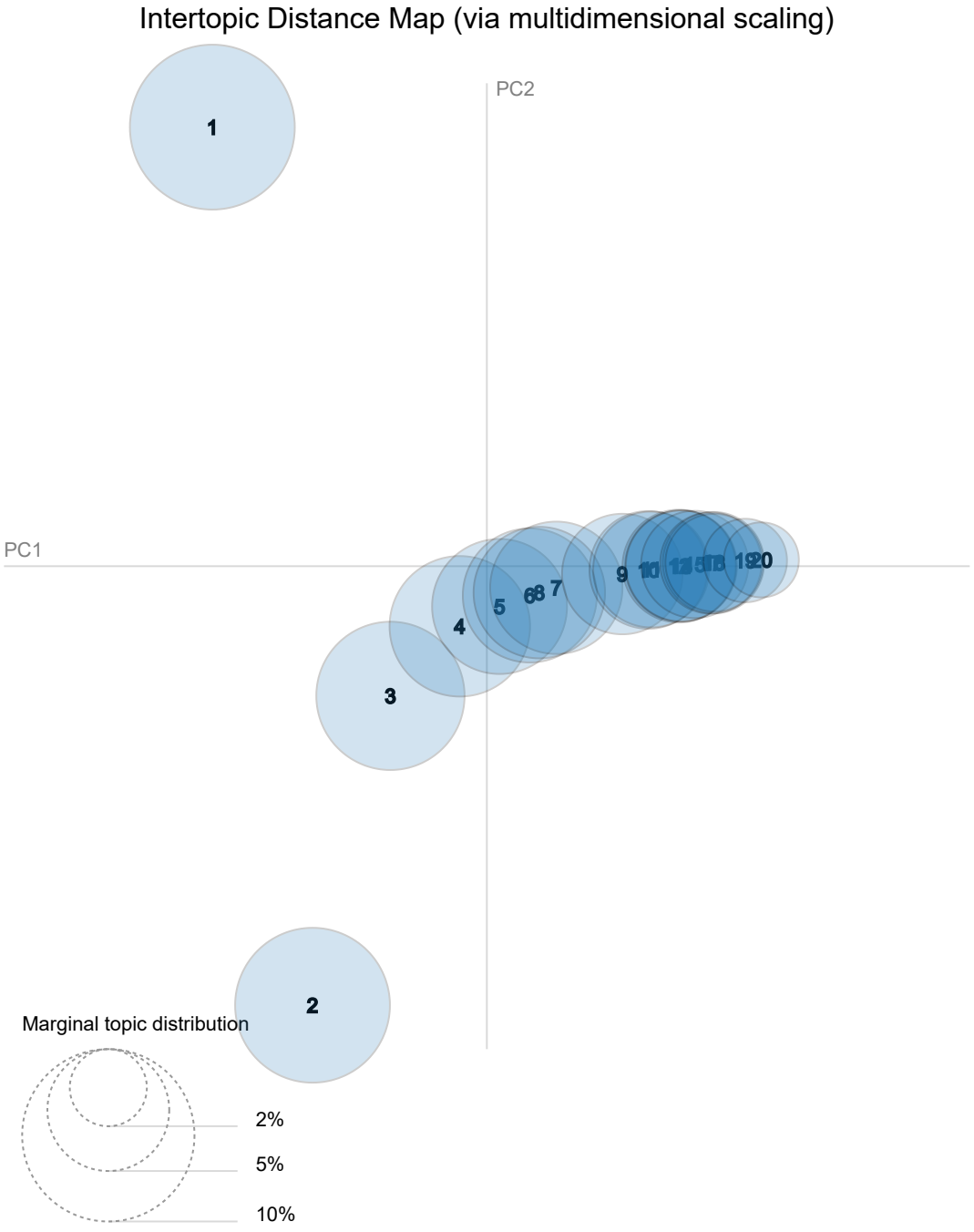
Selected Topic: 12 Previous Topic Next Topic Clear Topic



In [35]:
```
1 vis
```

Out[35]:

Selected Topic: 12 Previous Topic Next Topic Clear Topic



In [31]:

```
def format_topics_sentences(ldamodel=None, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row_list in enumerate(lda_model[corpus]):
        row = row_list[0] if lda_model.per_word_topics else row_list
        # print(row)
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = lda_model.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.Series([int(top
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Top

    # 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)


df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model, cor

# Format
df_dominant_topic = df_topic_sents_keywords.reset_index()
df_dominant_topic.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc
df_dominant_topic.head(10)
```

Out[31]:

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywor
0	0	15.0	0.1344	good, price, point, already, improvement, supp
1	1	19.0	0.1595	get, make, expect, be, internet, high, totall
2	2	15.0	0.1167	good, price, point, already, improvement, supp
3	3	15.0	0.1167	good, price, point, already, improvement, supp
4	4	12.0	0.2634	phone, return, first, still, amazon, disappoir
5	5	19.0	0.1090	get, make, expect, be, internet, high, totall
6	6	1.0	0.1837	battery, heat, awesome, drain, much, give, che
7	7	1.0	0.1292	battery, heat, awesome, drain, much, give, che
8	8	6.0	0.1940	problem, hang, service, thank, customer, guy,
9	9	3.0	0.0791	product, work, well, excellent, properly, supp

In [36]:

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

cols = [color for name, color in mcolors.TABLEAU_COLORS.items()]  # more c

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

topics = lda_model.show_topics(formatted=False)

fig, axes = plt.subplots(2, 2, figsize=(10,10), sharex=True, sharey=True)

for i, ax in enumerate(axes.flatten()):
    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')


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


Topic 0
worth
show
big
really
software
receive
support
range
awsome
always
Topic 1
featu
touch
miss
properly
go
gre
software
Topic 2
totally
be
make
else
high
tell
discharge
get
internet
expect
Topic 3
happen
time
lot
hour
half

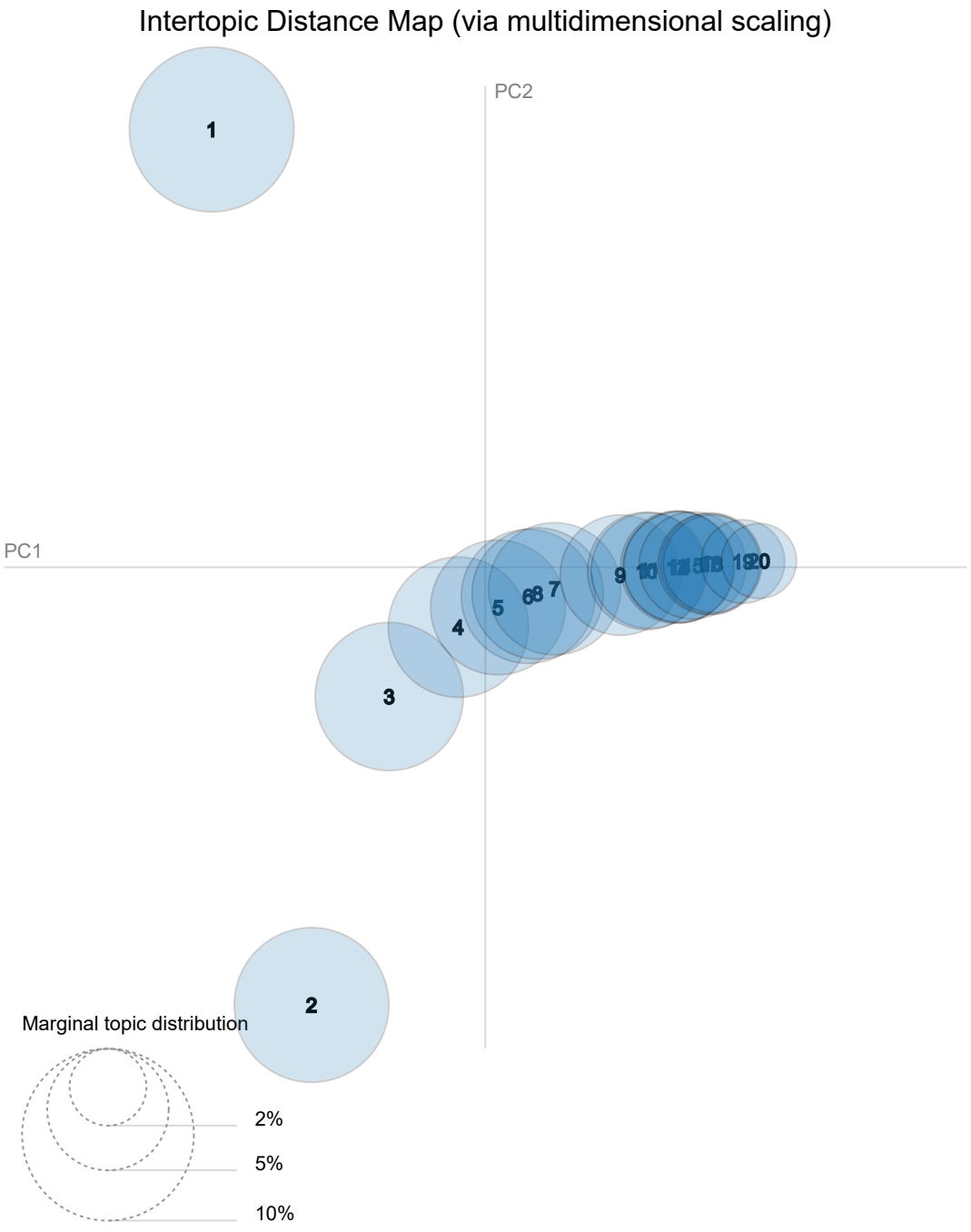
In [43]:

```
import pyLDAvis.gensim
pyLDAvis.enable_notebook(local=True)
vis = pyLDAvis.gensim.prepare(lda_model, corpus, dictionary=lda_model.id2
vis

```

Out[43]:

Selected Topic: 12 Previous Topic Next Topic Clear Topic



In [ ]:
```
1
```

In [ ]:
```
1
```